学校代码：10406
分类号：TN919　　学号：120085208001

南昌航空大学

硕 士 学 位 论 文

（专业学位研究生）

基于 Zigbee 的智能家居系统设计

硕士研究生： 吴　荔

导　　师： 宋高俊

申请学位级别： 硕　　士

学科、专业： 电子与通信工程

所在单位： 信息工程学院

答辩日期： 2015 年 6 月

授予学位单位： 南昌航空大学

The Design of System for Smart Home Based on Zigbee

A Dissertation

Submitted for the Degree of Master

On Electronic and Communication Engineering

By Wu Li

Under the Supervision of

Prof. Song Gaojun

School of Information Engineering

Nanchang Hangkong University, Nanchang, China

June,2015

摘 要

中国经济社会改革取得巨大成就的同时，人们对舒适、便捷而安全的智能型家居生活提出了要求。然而，总是随着新技术而不断发展的智能家居一直没有一个规范的技术标准。当前，一批新兴的无线传感网络技术给探索中的智能家居企业带来了新的曙光。随着国家在战略层面对智能家居进行规划与支持，一大批新的智能家居公司涌现出来。特别是传统家电企业与互联网科技巨头结盟的模式，频繁见诸媒体。火爆的智能家居概念已被推至业界瞩目的风口。

本论文在对国内外智能家居的发展概况和现状做出分析后，提出一种基于 Zigbee 无线传感网络的智能家居系统设计方案。作为时下热门的物联网技术，Zigbee 有着一般无线传感网络技术不可比拟的低功耗和大规模组网能力，非常适合智能家居的应用。论文对 Zigbee 技术特点和通信协议进行详细阐述后，对其硬件电路和组网通信程序进行了设计。为了将家庭无线传感网络与因特网建立连接、实现远程控制，论文对家庭网关的硬件以及网络通信协议的移植进行了详细设计。之后，论文选择当前市场占有率第一的安卓智能手机作为监控终端，并根据“控制设备、获取信息和信息交互”三种监控模式设计了相关功能的安卓监控软件。同时，论文还对实现家庭传感网络和手机端信息通信的服务器端进行了相关软件设计。最后，依托中国科学院深圳先进技术研究院机器人重点实验室的条件完成了相关软硬件开发与调试，并对系统的各个模块及整体进行了测试。测试结果表明各项设计指标基本实现，证明了系统设计的合理性。

关键词： 智能家居， Zigbee， 家庭网关， 远程控制， 安卓

Abstract

China has made great achievements in economic and social reforms, while the requirement of a comfortable, convenient and safe household life of intelligent for people was put forward. However, there has not been a normative technical standard as the smart home always improved with new technology. At present, a number of emerging wireless sensor network(WSN) technologies have brought a new dawn to the enterprises which are exploring in the field of smart home. With the country making more plans and support for smart home at the strategic level, a large number of new companies emerged. Especially, the traditional home appliance enterprises and Internet technology giant alliance mode, frequently reported by the media. The popular concept of smart home has been pushed to the attention of the industry.

After making the analysis about the history profile of development and the present situation of domestic and foreign smart home, this paper put forward a kind of smart home system design based on wireless sensor network of Zigbee. As nowadays popular technology of internet of things, Zigbee is ideal for smart home application with its incomparable low-power and large-scale networking capability. The paper presented the characteristic and the communication protocol of Zigbee in detail, and then designed the hardware circuit and the network program. In order to connect the WSN and the Interenet to realize the remote control, this paper made detailed design on the hardware of home gateway and the transplant of the network communication protocol. After that, the paper selected the current market share of the first android smart phone as the monitoring terminal. And designed the android software of related functions according to the three kinds of monitoring mode-control equipment, access to information and information interaction. At the same time, the paper also designed the sever software which aims at realizing the communication between home sensor network and the monitoring terminal. Finally, the related hardware and software development and debugging was completed relying on the advanced condition of key laboratory of robot in Shenzhen Institute of advanced technology, Chinese academy of sciences. Test results shown the realization of basic design criterion and proved the rationality of the system design.

Key Words: Smart home, Zigbee, Gateway, The remote control, Android

目 录

第 1 章 绪 论

1.1 选题背景及意义

世界上第一栋智能型建筑案例是位于美国 Hartford 的 CityPlaceBuilding。这是美国联合科技公司（United Technologies Building System）融合了信息化和整合化概念而设计的，它的出现点燃了世界各国竞相发展智能家居的热情[1]。

智能家居，是将各种先进的传感技术、控制技术、通信技术融合到家居电器与设备，使其形成一个彼此协同工作的智能化系统，从而构建一个安全、舒适与便捷的居住环境。

起初因为传感与网络技术的局限，总线技术和电力线载波通信技术是智能家居的主要技术。但是有线的技术有着先天的不足，如安装布线十分不便、检测维修困难、可扩展性差和施工成本高等问题，用户体验度非常不好。技术的局限性限制了智能家居的广泛使用，人们更多只在概念层面对未来智能家居进行设想。

近些年，微机电系统、片上系统、无线通信和低功耗嵌入式技术的飞速发展，催生出多种无线传感器网络（Wireless Sensor Networks，WSN）技术。它们凭着低功耗、低成本和自组织等特性给信息感知世界带来了新的曙光[2]。无线技术的出现满足了人们对自由的向往，它的主要优势在于无需重新布线，安装方便灵活。其良好的可扩展性和易改装性，于新装修用户和已装修用户都很适用。它的出现解决了拆墙布线的麻烦，从而节省了时间、人力、物力成本，同时使故障检测、设备替换等后期维护变得非常容易。

另一方面，在万物互联的发展格局下，不断发展的移动互联网技术和物联网技术以及宽带等基础设施的升级，促使智能家居产品联网化成为趋势。结合身边悄然兴起的大数据、云计算等各种科技，以及各种智能终端的大规模普及，智能家居必将使人们的生活变得更加舒适、便捷[3]。

在科学技术迅猛前进，人类逐步迈入智慧化时代的今天，人们对生活品质的追求也越来越高。智能家居已经不再单单停留在人们的憧憬之中，它已经成为一颗耀眼的明星，使人们的生活变得眼前一亮。致力于提升家居生活的品质与追求，打造便捷与舒适的家居生活新体验是本课题研究的意义所在。

1.2 国外智能家居发展概况

智能家居不是一个个孤立的家居产品，而是大到电视、冰箱，小到开关、灯泡的各种设备的智能系统体系。通过设备的协调工作，为居家生活带来舒适和便捷。然而，环看全球，欧美发达国家的智能家居无论是在成熟度还是系统研发力度方面都一直处于领先地位。如美国的谷歌、苹果公司，韩国的三星、LG 公司，日本的本田、丰田公司等一大批国外知名公司先后跻身智能家居领域，致力于最新技术的家居智能化开发[4]。凭借多年的研发与应用经验积累、充足的研究投入以及先进的科学技术，国外智能家居已经逐渐发展成成熟的智能家居生态圈。

智能家居的发展大体上可以按照三个阶段来划分[5]。第一个阶段是面向单个电器设备，各个设备孤立工作没有联系的家庭电子化阶段。第二个阶段是面向功能的住宅自动化，一些电气设备可以通过组成不太复杂的网络以完成特定的功能。第三个阶段是面向家居系统设计的家居智能化阶段。它将各种家居设备整合到一个网络中，通过一个中央控制设备对其进行集中有效的监控，使家居设备的工作与环境和人的需要相协调。

回顾近一年来国外几大公司在智能家居领域的布局，便能感受到智能家居已成为各大科技巨头争相竞逐的重要领域。2014 年 1 月谷歌高价收购 Nest 和 Dropcam 两家智能家居公司的消息，引爆了全球智能家居市场的热情，谷歌也因此跻身智能家居的前沿领域[6]。苹果公司虽没有谷歌如此规模的并购，却也是步步为营，开展了一系列的动作。先是在苹果全球开发者大会上发布了 HomeKit 智能家居平台，并通过对 HomeKit 平台硬件规格标准的定制，很快带动了一批厂商发布与 HomeKit 兼容的设备。然而苹果并不满足只提供 HomeKit 平台，而是进而组建自己的硬件团队，打造自己的智能产品，如 AppleWatch 的发布。2014 年 8 月，另一科技巨头——三星公司也不示弱，通过收购 SmartThings，获得一个成熟的家庭自动化平台。吸引三星的，是 SmartThings 支持别的厂商开发支持其生态链的应用，这种开放性能推动三星在智能家居领域的创新[7]。

总体来看，国外发达国家的智能家居由于起步早，技术与应用经验丰富，处于比较成熟的阶段。在引领智能家居先进技术的同时，也缺乏一个统一的国际标准。实力强的公司通常都开发自己的相关技术协议，彼此之间不能很好的兼容。

1.3 国内智能家居发展概况

随着中国经济社会的快速发展，人民的生活水平显著提高。人们对居家生活品质的日益重视，使得拥有一个更加舒适、便捷的智能型家居的需求变得日益迫切。然而，由于智能家居在我国的起步时间很短，在较长的时间内都一直处于探索阶段，同时，技术的落后与创新不足一直制约着智能家居的发展[8]。

今天，随着中国科技实力的积累以及无线传感网络技术的兴起，越来越多的中国企业投身物联网智能家居产业，催生出一大批有实力有创意的智能家居企业和产品。同时，政府的重视和媒体的持续热潮，使得智能家居概念成为公众关注的一个焦点。

智能家居的实现基础在于网络。近年来，国家对网络基础设施的政策与投入，包括要求新建小区宽带必须实现光纤入户以及 4G 高速网络的兴起，对智能家居将来的大规模普及打好了基础[9]。

此外，政府更在战略层面对智能家居进行规划，如工信部发布的物联网 “十二五”发展规划，把智能家居列入 9 个重点领域应用示范工程[10]；工信部、发改委等 15 个部委联合发布的《物联网发展专项行动计划》中，“推动智能家居应用”被列为重点任务[11]。智能家居在政府政策的大力扶持下，在 2013 年进入了快速发展阶段，显示出广阔的市场发展前景。《2012-2020 年中国智能家居市场发展趋势及投资机会分析》预测，中国智能家居市场在 2012 年至 2020 年间的年平均增长率将达到 25%左右，潜在需求高达数千亿[12]。

如今的智能家居已经被推到风口，国外大公司竞相圈地的同时，国内各大传统和互联网巨头公司也纷纷通过结盟、并购的方式布局智能家居市场。过去的一年以来，许多的大型企业间进行了战略合作，如 360 与 TCL 集团联手推出智能空气净化器、与奥克斯达成智能空调的战略合作；小米战略入股美的；魅族向海尔开放 lifekit 权限换取海尔对其开放 U+SDK；海尔集团与恒大集团在家电、家居等领域开展战略合作；TCL 集团同万达集团签署涉及智慧家庭的战略合作协议等等。

虽然近十年的发展使得中国智能家居获得了较大的发展，甚至部分品牌的智能家居产品获得出口订单，但是，我们要看到自身的发展还有许多不如意的地方，需要我们去重视和改善。首先，与日常生活必需品相比较，目前的智能设备虽然可以一定程度带来便捷，但却显得可有可无，寻找“刚需”入口是突破口。其次，价格不够“亲民”。目前市场上的一个智能灯泡或智能插座可能是普通灯泡或插座价格的几倍至几十倍，要让大众用户接受这样的价格需要时间。再次，当前智能家居业内没有统一的标准，各大公司都自成体系，各自的产品互不兼容，要让用

户选择所有家电的设备使用单一品牌是不容易的。最后，用户普遍担心的一个问题——安全与隐私。一旦智能设备接入网络，就存在家庭隐私和住宅安全遭到侵犯的可能。智能家居要得到消费者的认可就需要解决这些问题，因为消费者需要的是用简单的方式使用产品从而享受舒适的生活，而不是名义上“智能”，实际操作时却要很多甚至复杂的准备工作，同时还要付出较高的成本和安全代价。

1.4 论文主要内容及安排

本文对时下热门的智能家居的国内外发展概况和现状进行分析后，提出了一种基于 Zigbee 无线传感网络的智能家居系统设计方案。论文主要对系统中的四大块进行了详细的设计：Zigbee 无线传感网络、家庭网关以及服务端和客户终端。首先对 Zigbee 无线通信技术特点及通信协议进行了详细阐述，并且对其硬件电路和组网程序进行了详细设计。然后对家庭网关的相关硬件及网络通信协议的移植进行了详细的设计。之后，对安卓手机终端监控软件的开发及与服务端 socket 通信进行了详细的设计。最后，依托中国科学院深圳先进技术研究院的机器人重点实验室的平台完成了论文的相关实验与测试。

论文正文安排如下：

第一章介绍了课题研究背景和意义、国内外发展的现状以及主要内容的安排；

第二章主要对智能家居系统总体方案进行了设计；

第三章介绍了 Zigbee 的技术特点及通信协议，同时对传感器和通信节点的硬件电路及组网通信程序进行了设计；

第四章对家庭网关的硬件及通信协议的移植进行了设计；

第五章对 TCP 通信协议进行介绍后，对客户端和服务端软件进行设计；

第六章主要是对系统各个部分及整体进行测试；

第七章是对论文工作的总结和对未来工作的展望。

第 2 章 智能家居系统方案设计

2.1 系统架构

本论文设计的智能家居系统结构框架图如图 2-1 所示，总体上分为四个部分：Zigbee 传感网络、网关、服务器端和移动终端。系统工作原理如下：

Zigbee 传感网络是由网关中的 Zigbee 协调器组建的无线传感网络，通过将网络中的 Zigbee 路由器或终端节点嵌入到各个家居设备，实现对设备的控制。所有的控制信号通过 Zigbee 协调器发送到各个 Zigbee 终端节点，Zigbee 终端节点收到信号后解析命令，并对家居设备进行相应操作，然后将操作结果反馈给 Zigbee 协调器。网关的另外一个功能是实现 Zigbee 传感网络和 Internet 的数据通信。其内部集成了以太网控制模块，可以实现网关与网络服务器建立连接。这样，远程控制信号可以通过 Internet 网络发送给网关，网关再将控制信号发送给 Zigbee 终端节点。同时，Zigbee 终端节点的反馈信息也可以原路返回到远程控制终端。控制终端选用已经普遍使用的安卓智能手机。通过在安卓手机上安装为智能家居相关功能设计的安卓控制软件，可以实现随时随地方便地对家居设备进行控制。要实现手机远程通过流量发出控制指令给家居设备，并且收到家居设备的状态信息，还需要有固定 IP 的服务器端。网关和智能手机通过连接到固定 IP 的服务端，实现数据的处理与中转，从而实现了手机控制端和家居设备的交互。

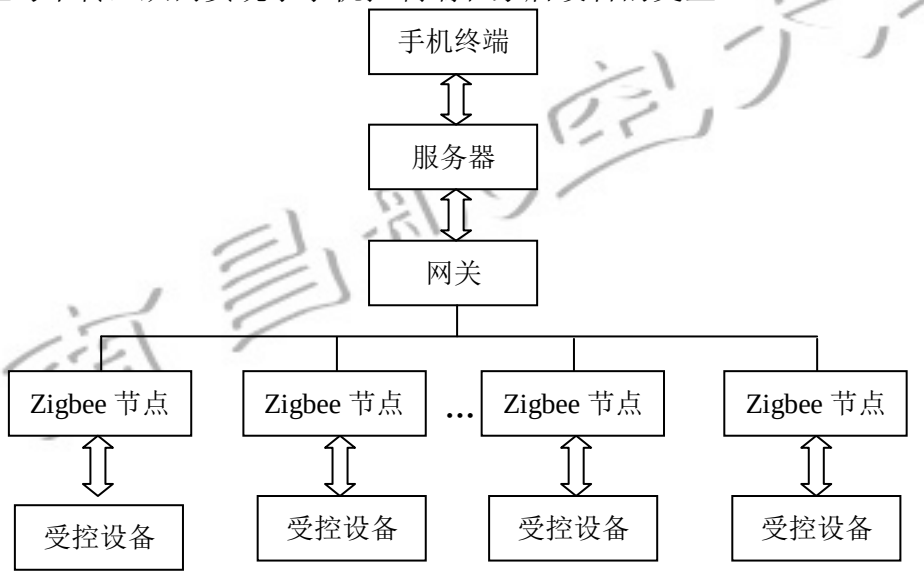

图 2-1 智能家居系统结构框架图

2.2 无线传感网方案

2.2.1 无线通信技术的选择

近些年，迅速发展的无线通信技术领域相继出现了多种无线网络数据传输标准，如 WiFi、蓝牙、Zigbee 等。不同的通信标准有着各自独特的优势，适用于不同的领域。以下是对这三种无线技术的介绍：

（1）Wi-Fi 技术

所谓 Wi-Fi，其实就是 IEEE 802.11b 的别称，是由一个名为“无线以太网相容联盟”（WECA）的组织所发布的业界术语，中文译为“无线相容认证”。之后更名为“Wi-Fi 联盟”的 WECA 组织通过制定一套认证标准，提出了 802.11 技术的一系列规范，包括 802.11g、802.11a/b、802.11n 等。现在 IEEE 802.11 这个标准已被统称作 Wi-Fi[13]。作为一种短距离、高速率通信技术，Wi-Fi 的有效信号覆盖范围半径达到上百米。相对于蓝牙，Wi-Fi 的数据安全性要差一些，但有着较快的数据传输速率，如 802.11n 传输速率达到 300Mbps,同时也具有较高的功耗。目前广泛应用于家庭、办公场所、公共及消费场所，为 Internet 的高速接入提供“热点”。

（2）蓝牙技术

蓝牙，是一项定义在 2.4GHz 频段、最大通信传输速率为 1Mbps 左右、通信距离为 10 米左右的短距离无线通信技术。蓝牙支持身份验证和数据加密，具有很好的抗干扰能力和多种低功耗模式[14]。在一个蓝牙网络中，能够和蓝牙主设备同时通信的蓝牙设备数最多只有七个。蓝牙技术常用于传输讯息和语言信号，广泛运用于数码产品、家用电气和可穿戴式传感设备。

（3）Zigbee 技术

Zigbee 是一种基于 IEEE802.15.4 标准的典型无线传感网络（WSN）通信技术，拥有 WSN 的诸多特点[15]。其中，最为突出的性能莫过于 Zigbee 的低功耗，测试表明，在省电模式下两节干电池就能够支持其工作时间达到半年以上。采用动态路由方式使其具有强大的自组网能力，最多可组成 65000 个子节点的大网，并支持组播和广播特性。其 20Kbps~250Kbps 的无线通信传输速率能够满足绝大多数的低速传输应用需求。100 米左右的通信距离在增加功率放大后可达到 1~3 公里，能够满足各种日常工作场所的需求。通过一系列的安全技术如物理层的扩频技术、MAC 层的应答重传机制，能够使 Zigbee 的数据传输可靠性得到保证。另外，可工作在三个 ISM 频段的 Zigbee 最多拥有 16 个通信信道，能够在网络受到环境干扰

时，动态地切换工作信道。目前 Zigbee 作为一种典型的无线传感网络，被广泛的应用在工业控制、环境监控和物联网方面[16]。

通过上面对三种无线通信技术的介绍，我们看到各种技术都有着各自的特点和使用领域。结合智能家居的住宅环境空间小、通信距离短、数据量小等特点，以及较少布线、较多接入设备等要求，Zigbee 技术无疑是非常适合的[17]。

2.2.2 组网方式的选择

Zigbee 拥有强大而灵活的组网能力，如图 2-2 所示，可由三种不同角色的 Zigbee 节点——协调器、路由器和终端节点，组成星型、树型和网状网三种网络结构。在具体开发时，要依据项目的自身特点选择合理高效的网络结构。

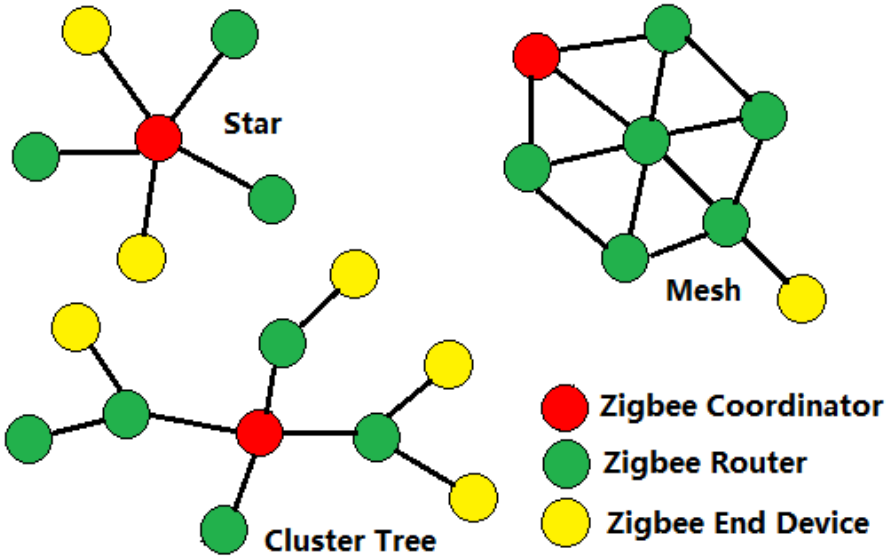

图 2-2 Zigbee 网络拓扑图

星形拓扑，就是所有的子节点围绕一个核心节点——协调器 Coordinator，形成发散式星形连接。显然，所有终端设备间的数据通信都需要依靠协调器的转发。而且，节点间的数据路由只有唯一的路径。一旦协调器不能正常工作，所有节点设备都不同通信。值得一提的是，由于 802.15.4 定义的底层 Zigbee 协议已经实现了这种网络形式，星形网络不依靠 Zigbee 网络层协议也能实现数据通信[18]。然而，用户必须得自己做更多应用层的工作，特别是对终端节点间数据通信的处理方法。

树形拓扑网络相对星形网络复杂，它允许路由器的加入，使得网络的拓扑形状看似分叉的树枝。在树形网络中，协调器 Coordinator 连接一系列的路由器 Router 和终端节点 End Device, 它的子节点也具有同样的功能，这样可以重复多个层级。与星形网络不同，树形网络的节点除了可以和它的父节点直接通信外，还能和自己的子节点进行通信，甚至通过路由的方式和较远的节点进行通信。与

星形网络相同的是，树形网络通信的路径也是唯一的。数据的传递方式是：先一直向上传递到自己的祖先节点，然后再向下路由传递到目标子节点[19]。数据传递的路由过程相对应用层来说是透明的，全部由协议栈操作。

Mesh 网状拓扑有着树形拓扑网络类似的结构形式，不单允许路由器的加入，而且还能通过路由节点发挥更强大的功能。路由器不单可以在有需要的时候接力传递数据，当遇到障碍物或者发生个别节点失效时，凭借 Zigbee 的动态路由机制可以通过找寻附近其他设备开辟新的传输路径[20]。它们的存在使得 Zigbee 网络具有强大的自组网和自愈能力。网络中的一个源节点向目的节点发送数据时，理论上可以通过其附近的任意一个对等路由节点经过多次接力完成。这时，数据的路由会有多个路径。但 mesh 网状网络能够选择最可靠的通讯节点形成一条成本效益最高的路径，这就是其“多跳”功能。这种“多跳”的功能同时也为其提供了一定的容错能力。即某一个中间节点失效时，网络可以通过其它的节点继续路由。

结合家居环境的复杂性，本次设计采用功能强大的 Mesh 网状网络。Mesh 网络可以很好的适应各种复杂家居环境，特别是多房间、多墙壁、上下层等通信障碍都可以得到解决。

2.3 家庭网关方案

2.3.1 网关结构

作为智能家居系统的核心部分，家庭网关是将家庭无线传感网络与外部 Internet 网络相连接，实现远程监控的关键要素。其内部主要由三部分组成：负责构建家庭无线传感网的 Zigbee 协调器、实现 Internet 连接的以太网控制器和核心微处理器 MCU。功能模块图如图 2-3 所示。

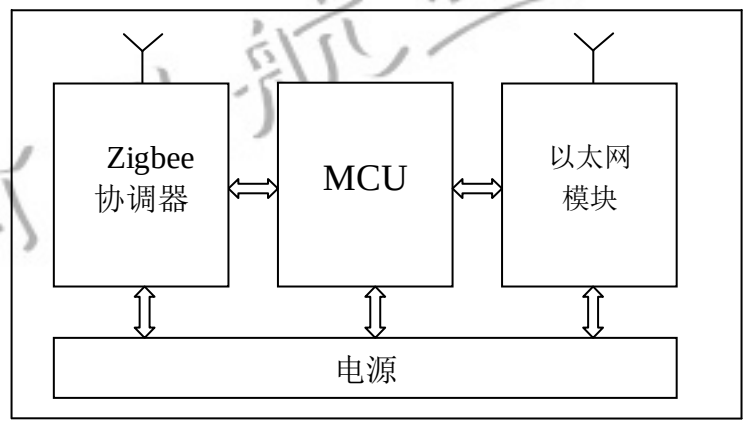

图 2-3 智能家居系统结构框架图

MCU 我们选择性能强大的 ARM 芯片。ARM 是英国一家芯片设计公司，主要提供 IP（Intellectual Property core）核，就是 CPU 内核结构，只包括最核心的部分，并不是完整的处理器[21]。ARM 公司自身不制造具体的芯片，而是出售芯片的设计授权给合作厂商。购买其芯片设计授权的主要是各大半导体公司，如 Philips 三星，ATMEL，甚至 Intel 等，它们拿到授权后再根据自身的强项设计并生产出具体的处理器芯片。一块完整的 ARM 芯片除了其内核，还需要很多其它组件。获得芯片设计授权后，芯片制造厂商就可以基于 ARM 内核增加许多外围设备如定时器、DMA、AD/DA 等。因此，各个的芯片制造厂商对于芯片的配置有不同的侧重。

以太网模块是负责与 Internet 建立连接的关键，有无线连接和有线连接方式。无线连接主要是通 GPRS/3G 模块，如 SIMCOM 公司的 SIM900A 芯片就是一款可以通过使用 AT 指令实现语音、SMS 和数据传输的工业级双频模块。有线连接方式则是通过网线接入因特网,依据 TCP/IP 等协议进行数据通信。

考虑到家庭环境中以有线方式接入路由器很方便，而无线方式受通信质量的影响且要增加额外的 GPRS 流量费用，我们选择有线方式。以太网控制芯片可以选择 ENC28J60、RTL8019AS、W5200 等性能优秀的芯片。

以太网模块结构如图 2-4 所示。

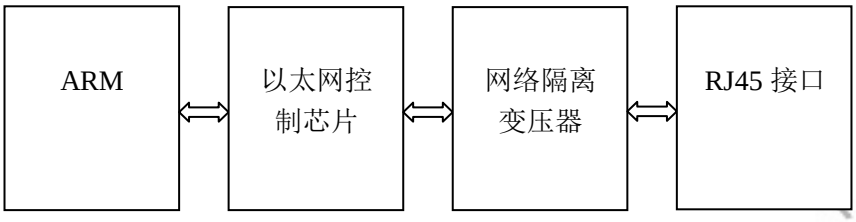

图 2-4 以太网模块功能模块

2.3.2 网络协议

选择以太网控制器作为有线连接网络方式后，网络通信协议我们可以选择 uIP 协议或者 LwIP 协议。例如，嵌入式常用的 uIP 协议算是精简版的 TCP/IP 协议，它在简化通讯流程的同时，最大限度的保留了必要的网络通信协议。我们选择其作为网关的网络通信协议。

uIP 是瑞典计算机科学学院开发的一个开源的 MVC 框架的协议栈。uIP 是针对嵌入式系统而开发的网络通信协议栈。它去除了 TCP/IP 协议中不常用的功能，只保留了必要的通信协议。uIP 协议栈通过简化网络通信流程，把网络层(Network Layer)和传输层(Transport Layer)作为设计重点[22]。使用 uIP 协议栈具有许多优点：

1） uIP 协议栈只有少量的代码，有利于查看和移植使用。

2） uIP 协议栈的运行只需占用很少的内存空间。

3） 多个层使用一个缓存区，既节省了内存空间又省去了数据拷贝的时间。

4） 允许多个主动/被动连接同时进行 。

5） uIP 提供通用性非常好的多个实例程序，十分便于开发者移植使用。

6） 无需操作系统即可实现事件的轮询处理[23]。

从以上 uIP 的众多优点可以看出，uIP 协议栈对资源的要求非常少，甚至大多数低端的单片机都能够轻松使用。也因此，它被广泛应用在许多的嵌入式设备之中。

如图 2-5 所示，uIP 在系统处于硬件底层和应用顶层之间，起到顶层与底层之间数据传递作用。uIP 提供一个 uip_input 函数供底层网卡驱动调用，一个 uip_periodic 函数供底层系统定时器调用，再通过 UIP_APPCALL 函数与应用层通信。作为一个由代码实现好了的库，它相对系统的内部协议组是透明的，具有很好的可移植性和通用性。

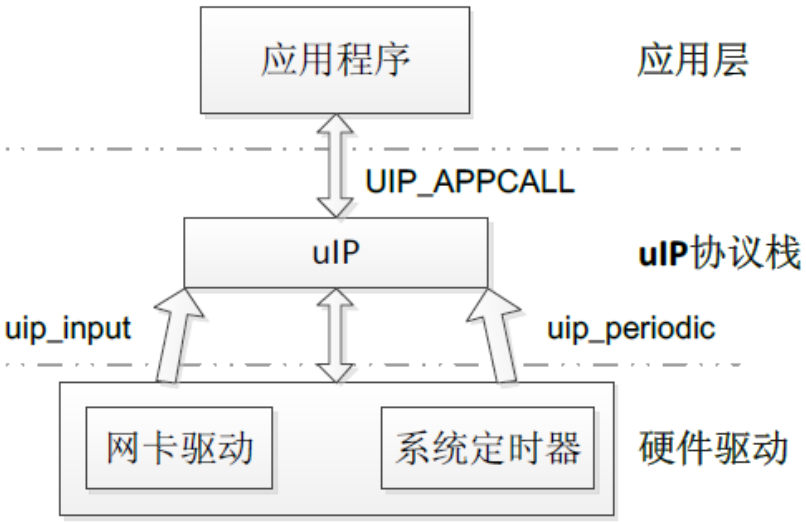

图 2-5 uIP 在系统中位置关系

由于不同的用户使用的 TCP/IP 情况不一样，需要开发者根据自身的情况编写自己的应用程序。为了提高开发者的效率，uIP 协议栈为用户提供了许多由 C 语言编写的宏命令作为可以直接调用的接口函数。开发者需要做的是将应用层入口函数定义为宏 UIP_APPCALL()暴露给 uIP 协议栈。当 uIP 收到底层发来的数据时，只要调用 UIP_APPCALL()就可以将数据发送给上层。这样就可以不改动协议栈而适用其它不同的应用程序。uIP 协议栈为开发者提供了许多以宏命令实现的接口函数，这样可以有效地降低调用函数时所需的额外开销。

第 3 章 Zigbee 传感网络设计

3.1 Zigbee 简介

3.1.1 什么是 Zigbee

自然界的蜜蜂（bee）在发现花粉后，会通过 ZigZag 字形舞蹈向远处的同伴传递花粉的位置、方向和距离等信息，以此构建群体的通信网络[24]。Zigbee 一词便来源于此。

Zigbee 是一项基于 802.15.4 标准的近距离、低复杂度、自组织、低功耗、低数据速率、低成本的无线通讯技术[25]。作为一种无线自组网技术标准，它可以组建最大容量 65000 个设备的网络。在具有极低功耗的强大性能之外它还有较快的响应速度。Zigbee 采用动态路由方式使它不单能够以路由的方式接力传递信息，从而突破点对点通信的距离限制，还能在遇到网络中若干节点失效的情况下，自动寻找附近其它节点重新规划传输路径，使其具有强大的自组网和自愈能力。

3.1.2 Zigbee 节点角色介绍

Zigbee 的逻辑设备类型有 Zigbee 协调器 Coordinator Device、Zigbee 路由器 Router Device 和 Zigbee 终端设备 End Device 三种[26]。它们各有分工，分别扮演着网络协调者、网络支持者、终端感知者三种角色。

作为权限最高的协调器 Coordinator Device，它是网络的创建者和管理者，在整个系统中最多只能有一个[27]。路由器设备 Router 的主要作用是扩展网络覆盖面进而支撑网络。它们既可以做协调器或路由器的子节点，又可以做其它路由器或终端节点的父节点。路由器可以在有需要的时候接力传递数据，并且当遇到障碍物或者发生个别节点失效时，凭借 Zigbee 的动态路由机制可以通过找寻附近其他设备开辟新的传输路径。它们的存在使得 Zigbee 网络具有强大的自组网和自愈能力。终端节点 EndDevice 位于网络的最末端，不能再有子设备。它们可以收到并反馈协调器或者路由器发来的信息，但不是不能再进行信息的路由操作。

3.1.3 Zigbee 协议及协议栈简介

通常，我们设计一个网络架构的一般做法是：不同的层，不同分工；同时，只允许数据在相邻的上下层之间流通。例如，由 ISO 组织提出的著名的 OSI 模型就是一个七层的分层体系架构，如图 3-1 所示。

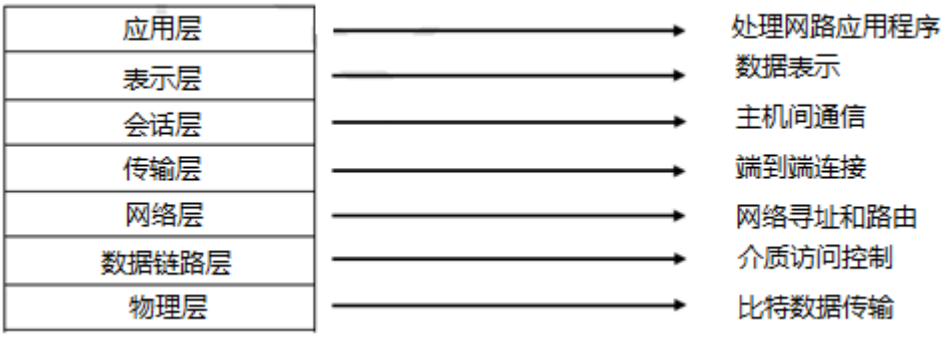

图 3-1 OSI 七层参考模型

Zigbee 通信协议虽然也采用了 Open System Interconnection 模型，但是只对与 Zigbee 有关联的层进行定义[28]。图 3-2 清晰的展示了其协议栈主要由两方面定义。其中，由 IEEE802.15.4 工作组定义的最底层的物理层和媒介访问控制层是协议栈其他层的基础。顶层的网络层和应用层则是由 Zigbee 联盟定义的，包括了应用支持子层等。基于 IEEE 组织的全球影响力，同时在 TI、ST 等知名 Zigbee 联盟成员的努力推动下，Zigbee 标准已经成为无线传感网络的不二选择。

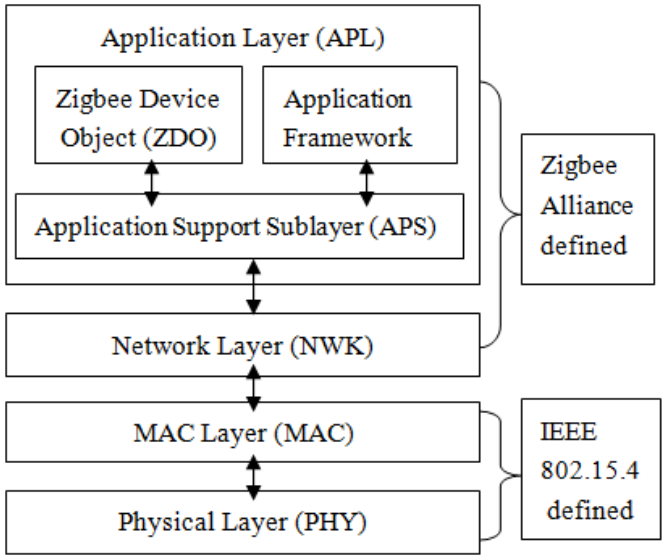

图 3-2 Zigbee 协议栈结构图

通信协议指的是通信的双方在进行正常通信时所依据的一个共同的标准。通信协议栈指的是通信协议的具体实现形式，通常是指已经用代码实现功能的方法，供开发者使用。Zigbee 协议栈就是开发者可以直接调用的应用层 API 接口，能够极大提高开发者的开发效率。有了协议栈，开发者在开发过程中不需要知道 Zigbee 协议的具体实现细节，只需要弄清楚关键的问题——数据从哪里来到哪里去。

3.2 Zigbee 节点硬件设计

论文对 Zigbee 核心板和底板进行独立设计，核心板将是以 Zigbee 芯片为核心的最小无线收发系统，底板是一些外围电路及不同的传感器。核心板和底板以插针的方式组合，模块化设计便于拔插及故障检测、替换。硬件的原理图及 PCB 设计将采用 Altium Designer 10 软件进行完成。

3.2.1 Zigbee 芯片选型

作为当前无线传感网络的一大热门技术，Zigbee 受到了各大芯片制造商的热捧，各自推出有自己特色的芯片以及开发平台。目前市场上主要的 Zigbee 芯片提供商有 TI、Ember、Freescale、Jennic 等，通过对比我们选择了 TI 公司的 Zigbee 芯片，主要基于以下考量：首先，TI 公司的 Zigbee 芯片是采用我们非常熟悉的增强型 8051 处理器，而其他厂商都是用自己的处理器；其次，TI 提供了免费的配套 Zigbee 协议栈，而其他许多公司的协议栈都收费；TI 的 Flash+MCU+ZigbeeRF 是真正的单芯片解决方案，而其他许多公司都需要额外的芯片。

德州仪器公司推出的 CC2530 芯片，如图 3-3 所示， 是用于 Zigbee 应用的一个真正的片上系统（SoC）解决方案。该芯片只需要很少的开销就能够工作，同时，多种运行模式使得它能够以超低功耗组建起功能强大的无线传感网络。值得一提的是，极短的模式切换时间使其进一步降低了功耗。CC2530 融合了先进的 RF 性能，普遍使用的增强型 8051 处理器以及 8KB 的可编程闪存，构成了 Flash+MCU+ZigbeeRF 模式的单芯片解决方案。本次设计我们选用 CC2530F256 型号芯片，它能够完美的支持 TI 公司业界领先的 Z-Stack 协议栈。

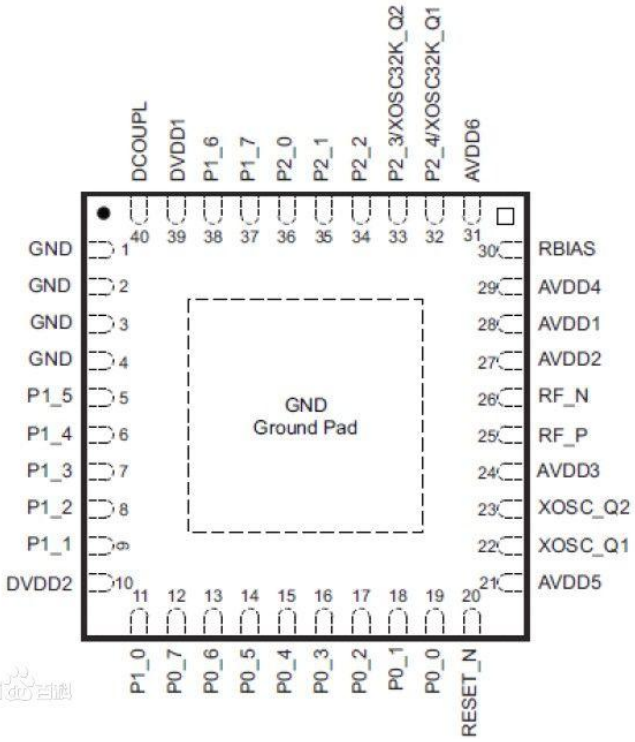

图 3-3 CC2530 芯片

除此之外，CC2530F256 还有强大的外围设备[29]：

–强大的 5 通道 DMA

–8 位和 16 位的定时器

–IR 发生电路

–硬件支持 CSMA/CA

–支持精确的数字化 RSSI/LQI

–电池监视器

–温度传感器

–模数转换器

–安全协处理器 AES

–2 个 USART 串行收发模块

–21 个通用 I/O 引脚

–看门狗定时器

3.2.2 Zigbee 核心板硬件设计

Zigbee 核心板，即让 CC2530 芯片能够工作的最小系统电路，主要包括晶振电路、滤波电路、电源去耦电路、阻抗匹配电路、巴伦电路等。如图 3-4 所示：

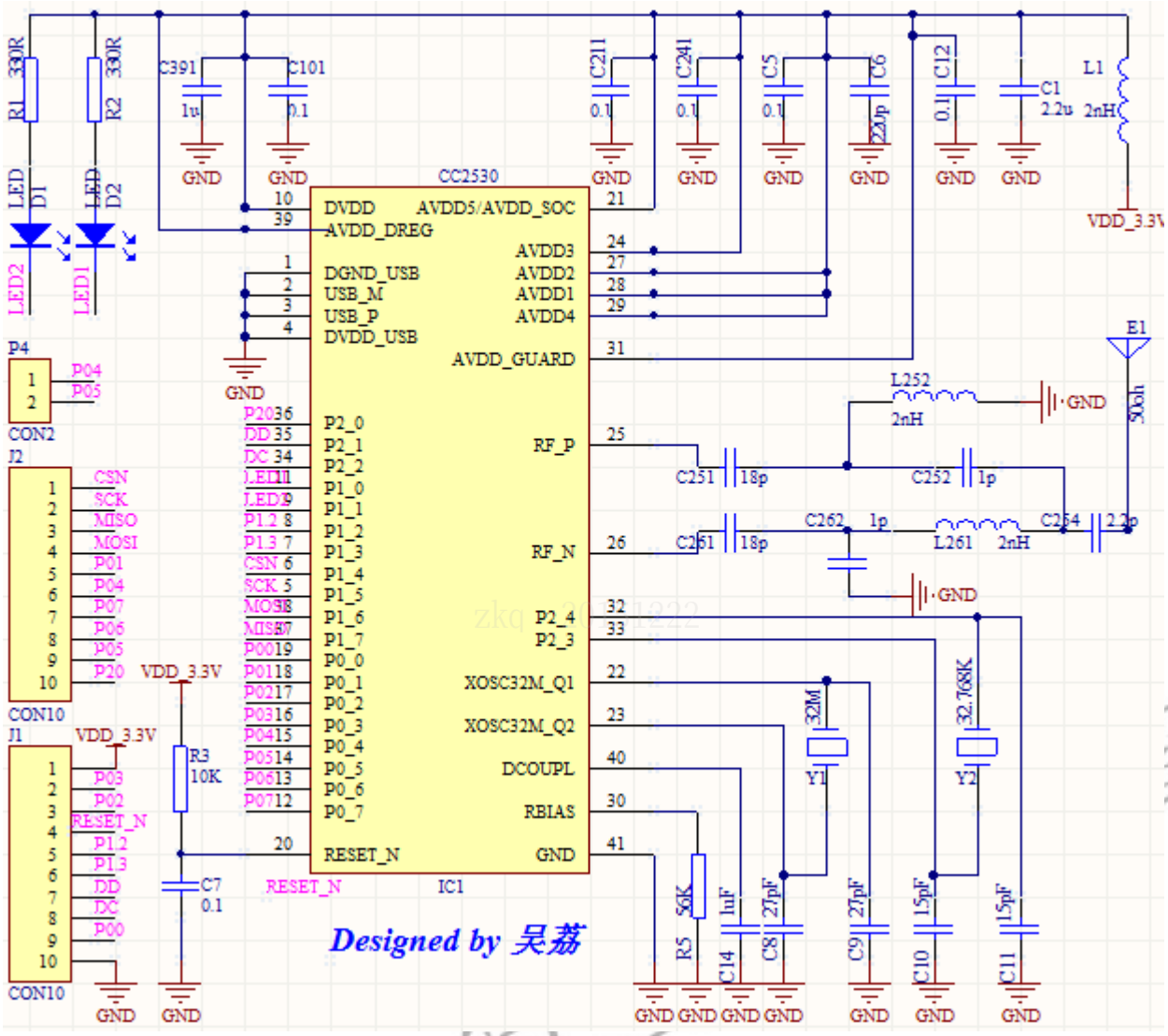

图 3-4 Zigbee 核心板硬件设计原理图

（1）天线及巴伦配置电路设计

用于发射和接收电磁波的天线是无线通信中的一个重要的部分。收发天线就是电磁波的出入口，是对传输线上的导行波和无界媒介中的电磁波进行切换的变换器件。作为发射机的原理是，通过馈线传递到发射天线的高频电流被天线转换成电磁波，向一定的方向发射出去。而作为接收机的的原理刚好相反，接收天线将从一定方向传来的电磁波变换成高频电流后送给输入回路。

天线通常有单端和差分之分，单端常被称作不平衡天线，反之，差分就称作平衡天线。巴伦是平衡不平衡转换器的意思，它的英文名字 balun 来源于单词“balanced”和“unbalanced”[30]。容易知道，balance 就是平衡的意思，代表差分结构；而 unbalance 是不平衡的意思，代表单端结构。由于 CC2530 芯片是差分设计的 RF 口，本次设计的天线属于单端天线，因此需要采用巴伦电路。

根据上面的天线理论知道，如果将属于“unbalanced”传输线的同轴电缆直接接上属于“balanced”的天线，其外皮的高频电流会对天线的发射与接收产生严重干扰。解决这个问题就需要在这两者之间加入巴伦，将同轴电缆表皮的高频电流滤除干净[31]。

通过上面的分析知道，在基于 Zigbee 网络的无线通信过程中，天线及巴伦匹配电路的设计非常重要，这涉及射频通路指标是否优良，直接影响通信质量以及系统功耗。图 3-5 为本次巴伦匹配电路原理图设计。

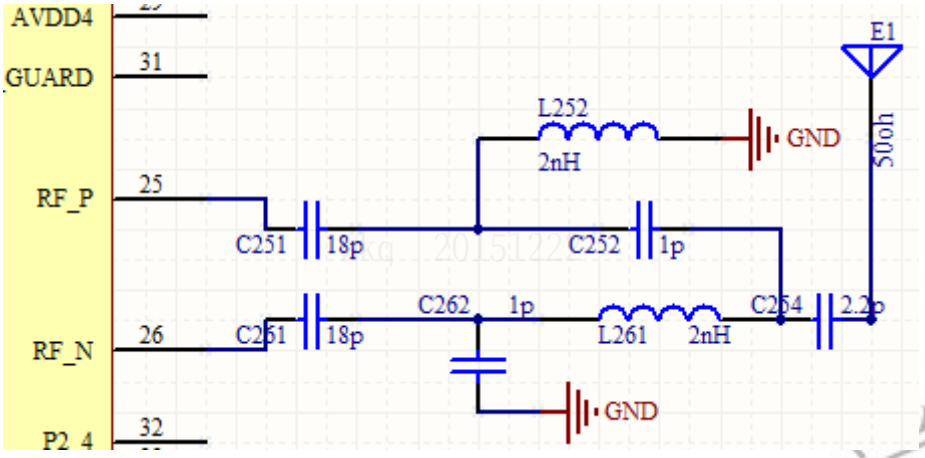

图 3-5 巴伦天线原理图设计

射频电路中的天线有很多种，最常见的有 PCB 天线、芯片天线和鞭状天线。在选择天线时，大小、成本、性能，都是重要的参考因素。图 3-6 展示了此三种天线各自的优缺点。

天线类型	优点	缺点
PCB天线	• 低成本 • 可能有很好的性能 • 高频条件下天线尺寸可能小	• 很难设计小型的高效PCB天线 • 低频时可能需要大尺寸
芯片天线	• 小尺寸	• 性能中等 • 成本中等
鞭状天线	• 好性能	• 高成本 • 在很多应用中很难适用

图 3-6 不同类型天线的优缺点

综合考虑三种天线的优缺点后，本次设计选择采用 PCB 天线和鞭状天线两种。鞭状天线用于需要性能较好的网关协调器，PCB 天线用于对尺寸要求较高的调光灯。PCB 倒 F 天线如图 3-7 所示，它能够非常小巧的嵌入各种设备中。

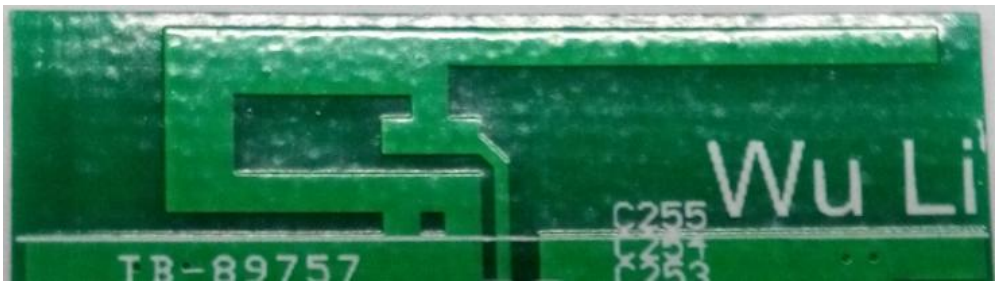

图 3-7 PCB 倒 F 天线

（2）电源去耦、滤波电路

为了在使用中获得良好的性能，电源去耦电容器的尺寸、布局和电源的滤波设计极为重要，这关系到系统稳定性以及无线发射距离和无线接收灵敏度。电源去耦及滤波电路如图 3-8 所示。

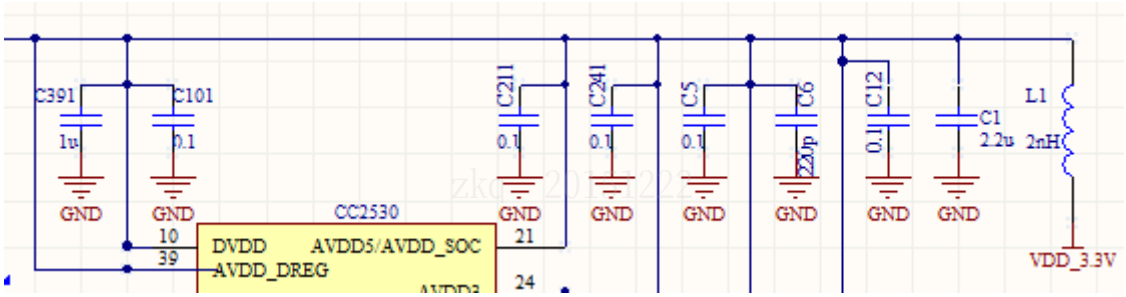

图 3-8 电源去耦及滤波电路

3.2.3 Zigbee 底板硬件设计

Zigbee 底板电路主要包括电源电路、JTAG 调试接口电路、串口电平转换电路等。

（1）底板电源电路

电源电路采用 DC-DC 变换器 LM1117 将输入的 5V 适配器电源转换为 CC2530 芯片所需的 3.3V 电源，如图 3-9 所示。LM1117 是一个低压差电压调节器系列，通过 2 个外部电阻可实现 1.25～13.8V 输出电压范围，也有 5 个固定电压输出（1.8V、2.5V、2.85V、3.3V 和 5V）的型号，这里我们选择 3.3V 型号。LM1117 还有多种保护措施，如芯片内部通过一个齐纳调节的带隙参考电压使输出电压的精度控制在±1%以内，此外，芯片还具有电流限制和热保护功能。为了改善输出电压瞬态响应和稳定性，常在芯片的输入输出端接上滤波电容，特别是输出端需要一个至少 10uF 的钽电容。

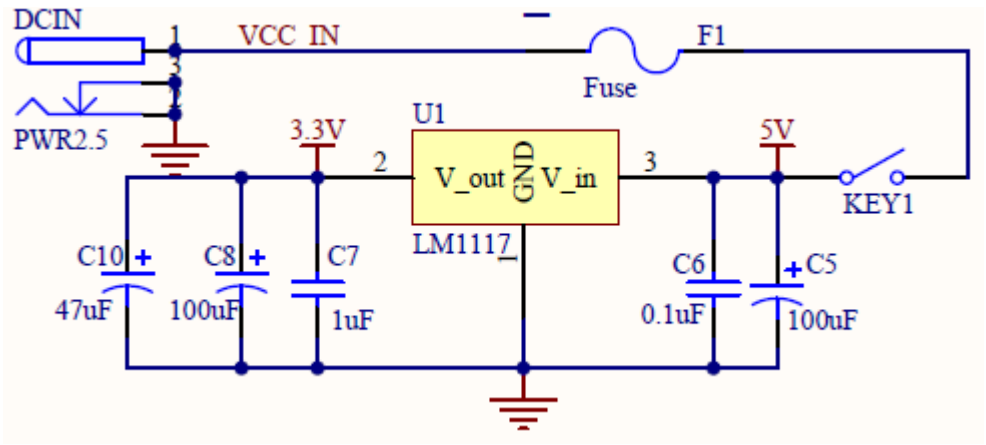

图 3-9 底板电源电路

（2）JTAG 调试接口电路

JTAG 调试接口是程序下载接口、仿真 SPI 接口和抓包工具连接端口。电路原理图如图 3-10 所示。

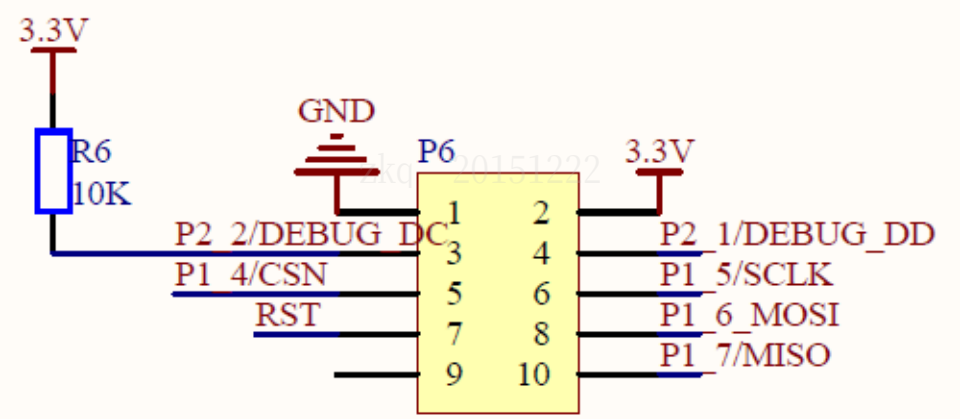

图 3-10 JTAG 调试接口电路

（3）串口电平转换电路

当需要 CC2530 芯片和计算机进行通信时，尽管 CC2530 芯片有串行接口，但它的信号电平和标准 RS232 的不相同，因此需要串口电平转换电路将 CC2530 串口的 TTL 电平转换为与计算机或者其他设备通信的 RS232 电平。串口电平转换芯片我们选择 Maxim 公司的 MAX232，它是一款专门为 RS232 标准串口设计的芯片。MAX232 通过芯片内部的电容性电压发生器，能够在 5V 电源供电时提供 EIA/TIA-232-E 电平。MAX232 串口电平转换电路如图 3-11 所示。

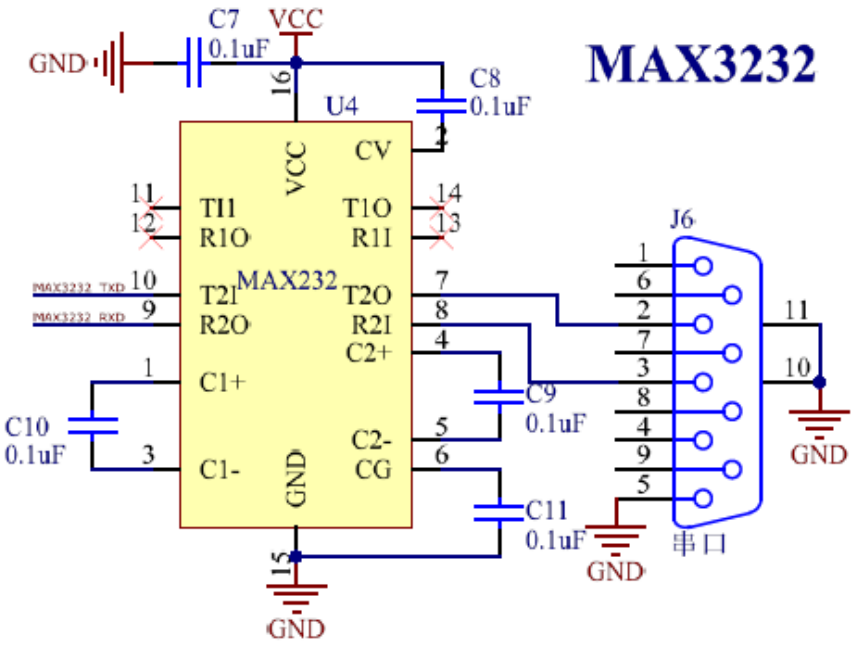

图 3-11 串口通信电平转换电路

3. 3 PWM 调光灯设计

3. 3. 1 PWM 调光原理

LED 即发光二极管，是一种被广泛使用的能将电能转换为光能的发光元件。由于具有节能、环保、使用寿命长、体积小、亮度高、显色性高等优点，LED 的使用已经非常普遍，甚至取代传统的白炽灯。要实现 LED 的调光，通常最简单的做法是在 LED 回路中串接镇流电阻，通过调节电阻的阻值来改变 LED 的驱动电流，从而实现改变 LED 的亮度。此方法虽然简单，但有着明显的缺点，如电流稳定度不高，电阻发热导致用电效率低以及不便于自动控制等。为了实现 LED 驱动电流稳定且使用效率高，本次设计采用恒流驱动方式。设计框图如图 3-12 所示。

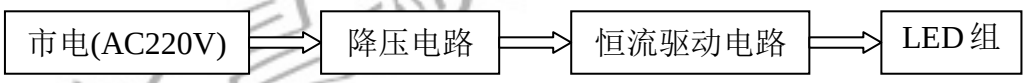

图 3-12 调光灯设计框图

这里的降压电路是将高压的交流市电变成低压的直流，我们选择使用常用的交流转直流适配器。为实现恒流驱动且自身功耗低，恒流驱动电路我们选择 PT4115 芯片芯片。为了保证芯片工作的可靠性，PT4115 内部具有过温保护电路。同时，采用 SOT89-5 的封装使得芯片内部的管芯连通到芯片外部的金属板，其散热性能

非常出色。此外，PT4115 的开关频率采用的抖频技术是一种分散谐波干扰能量有效改善 EMI 问题的新方法。PT4115 是一款连续电感电流导通模式的降压恒流源，内置功率开关，采用高端电流采样设置 LED 平均电流。其外部 DIM 引脚还可以接受 PWM 波的输入实现模拟调光。PWM，即脉冲宽度调制，是利用微处理器的数字输出来对模拟电路进行控制的一种技术。其控制方式是通过控制电路中电子开关器件的通断，使输出得到一系列幅值相等的脉冲，改变脉冲的宽度或占空比可以调节输出电压大小[32]。PT4115 芯片有着高达 5000 比 1 的调光比。

3.3.2 PWM 调光灯硬件设计

PWM 调光灯驱动原理图如图 3-13 所示，4 个二极管组成的桥式整流电路的作用是保证输出给 PT4115 的电流是直流。滤波电容 C1 的作用是将脉冲直流变换成平滑的直流；采样电阻 R1 的作用是控制芯片的输出电流大小；续流二极管 D5 的作用是在芯片内部 MOS 管截至时为电感提供放电回路。

PT4115 的 DIM 管脚用来接控制芯片的 PWM 波。现在的许多微处理器内部都带有 PWM 控制器，可直接选择接通的时间和周期从而方便的输出所需占空比的 PWM 波。若是微处理器内部没有 PWM 控制器，也可以使用定时器和 GPIO 口来模拟实现。

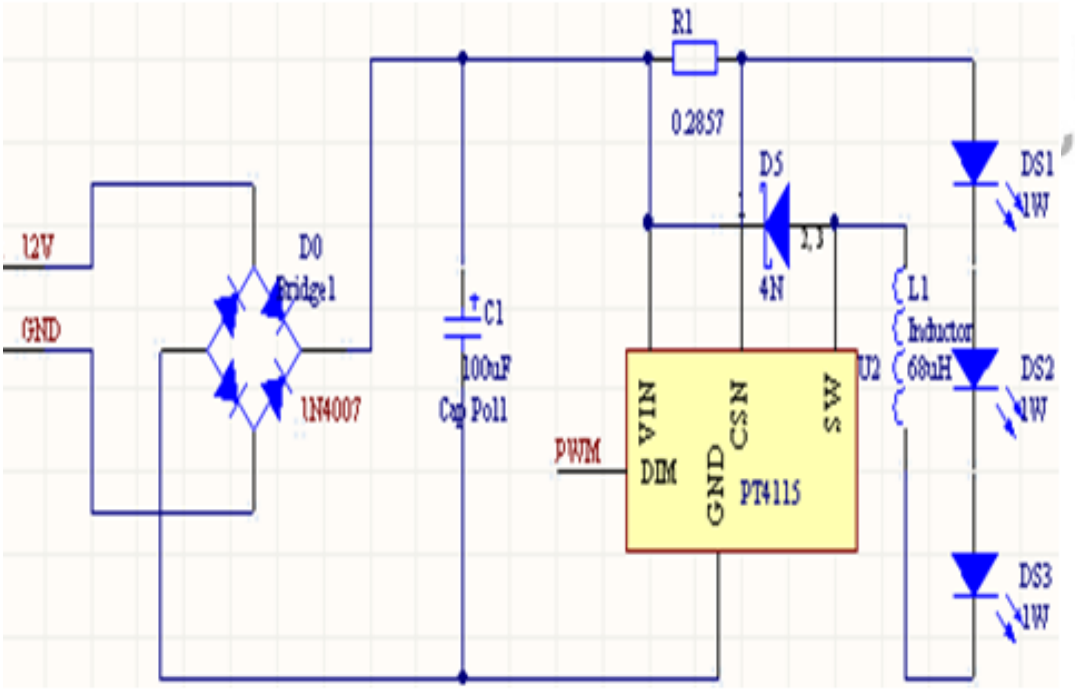

图 3-13 PWM 调光驱动原理图

3.4 温湿度模块设计

3.4.1 DHT11 简介

DHT11 传感器不单可以检测环境的温度值还可以检测湿度值，并且具备响应速度快、质量稳定可靠、价格低廉等特点。每个 DHT11 芯片都通过非常精确的校验，使其具有很高的数据可靠性。DHT11 芯片是一次性向外部发送数据，并且采用校验和的方式确保传输的可靠性[33]。

其实物图如图 3-14 所示，其关键性能参数如下：

工作电压范围： 3.3V-5.5V

工作电流： 平均 0.5mA

输出： 单总线数字通信

测量范围： 湿度 20~90％RH， 温度 0~50℃

精度： 湿度±5%，温度±2℃

分辨率：湿度 1%，温度 1℃

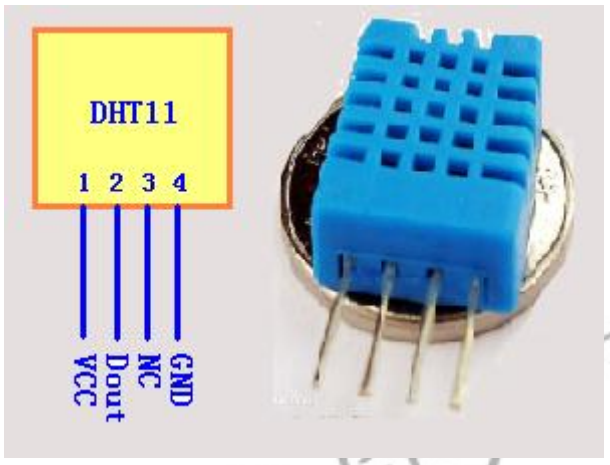

图 3-14 DHT11

3.4.2 温湿度模块硬件设计

如图 3-15 所示，DHT11 与 MCU 之间能采用简单的单总线进行通信，采集命令的发送和数据的读取仅仅需要一个 I/O 口就能够做到。其数据包由 5Byte（40Bit）组成，前四个字节依次是湿度值和温度值的整数和小数部分。第五个字节是校验和，由前 4 个字节数值相加而得。

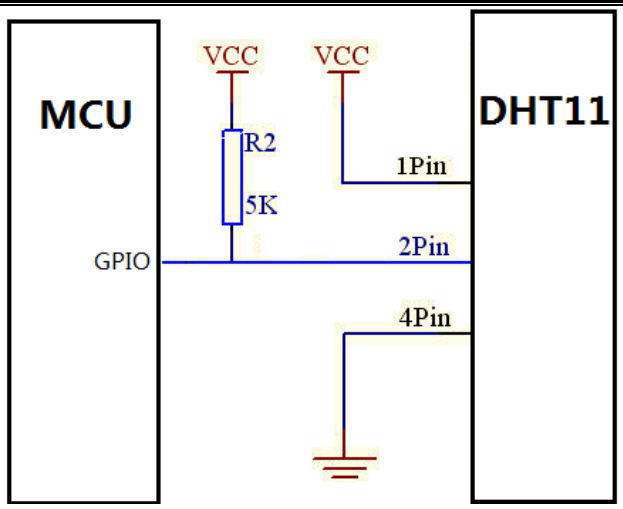

图 3-15 DHT11 原理图

由于 DHT11 是通过一根线一位一位的输出 40 位的未编码数据，处理器收到收据后要对数据进行分割处理。如某次从 DHT11 读到的数据为：0010100 00000000000011010000000001000010，对数据分析如下：

字节位：	byte4	byte3	byte2	byte1	byte0
二进制：	00101000	00000000	00011010	00000000	01000010
十进制：	40	0	26	0	66

由以上数据的分割分析容易得到温湿度值的计算方法：

湿度=byte4.byte3=40.0 (%RH)

温度=byte2.byte1=26.0 (℃)

校验=byte0=66=byte4+byte3+byte2+byte1 (校验正确)

3. 4. 3 温湿度模块程序设计

DHT11 的数据格式十分简单，和 MCU 的一次通信时间最大为 3ms 左右，连续读取时间间隔最好大于 100ms。如图 3-16 所示，读取数据时要严格按照 DHT11 的传输时序。

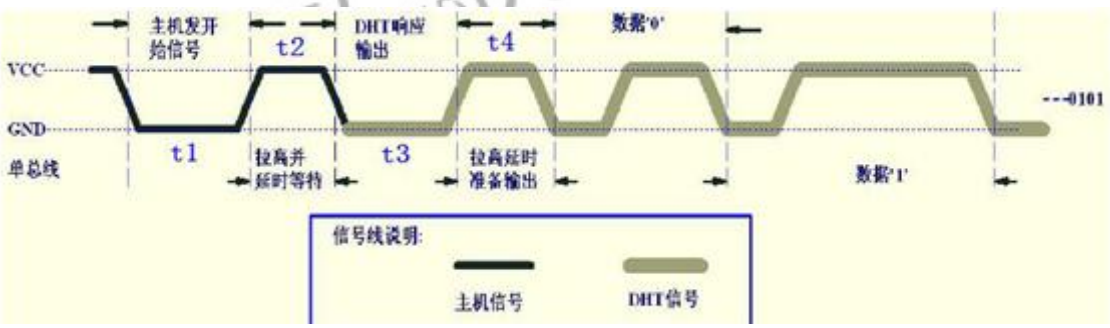

图 3-16 DHT11 数据读取时序流程

温湿度传感器的工作流程如图 3-17 所示，首先由处理器拉低数据线保持至少 18ms 时间，作为开始信号。之后，处理器拉高数据线并保持 20us 后让出数据线，等待 DHT11 的响应。当 DHT11 芯片收到了处理器的开始信号后，拉低数据线，保持 40us 左右作为响应。然后，再次拉高数据线，保持 40us 左右后输出温湿度数据。

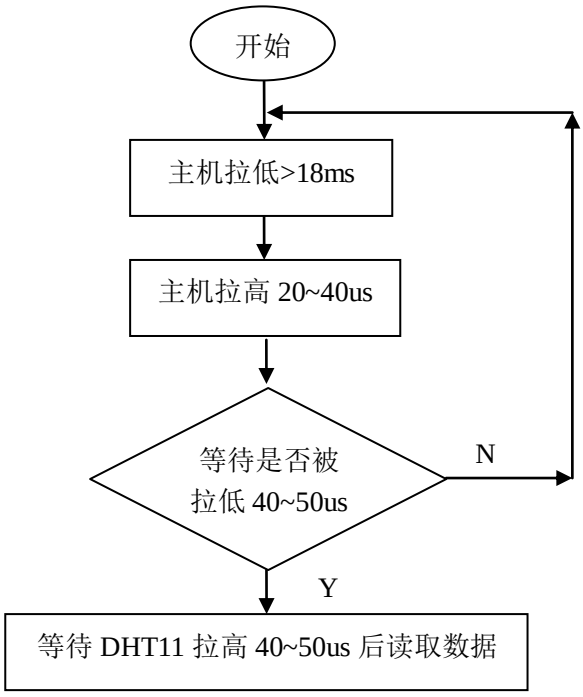

图 3-17 DHT11 数据读取流程图

3.5 Zigbee 组网程序设计

3.5.1 Zigbee2007 协议栈介绍

协议虽然是统一的，但协议栈却有很多版本，不同的公司、厂商提供的 Zigbee 协议栈有一定的区别，我们这里选择与 CC2530 配套使用的 TI 公司的 Zigbee 2007 协议栈 Zstack-CC2530-2.3.0-1.4.0。Zigbee 2007 协议栈又叫 Z-Stack，是由已被 TI 公司收购的 Chipcon 为其 CC2430 开发系统平台配套发布的一个优秀的 Zigbee 协议栈软件[34]。开发者可以通过 Z-Stack 轻松的开发出需要的应用，实现 Zigbee 无线网络通信。

Zigbee 2007 协议栈需要配合使用 IAR 公司的 IAR Embedded Workbench for MCS-51 作为集成开发环境。值得注意的是，IAR 和 Z-Stack 的高低版本是互不兼容的，二者的版本安装选取一定要配合好，我们选择的是 IAR 7.60A 和 ZstackCC2530-2.3.0-1.4.0 配合使用。

3.5.2 组网程序设计

德州仪器公司的 Zigbee2007 协议栈可以看作一个小型的操作系统，如图 3-18 的 IDE 目录所示，各个层的协议代码都分类编辑。和大多数程序一样，协议栈程序也是从主函数 main 开始运行的。主函数主要完成了两个任务：首先是系统初始化；然后是运行轮询式操作系统。如图 3-19 所示。

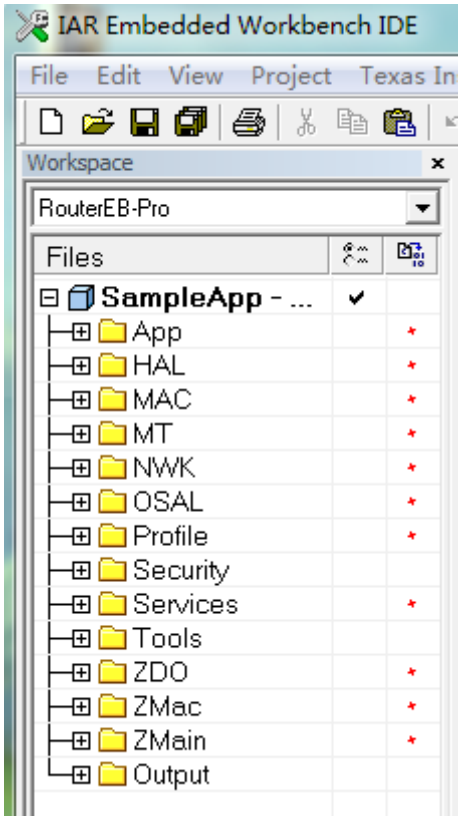

图 3-18 TI Z-stack-CC2530-2.3.0

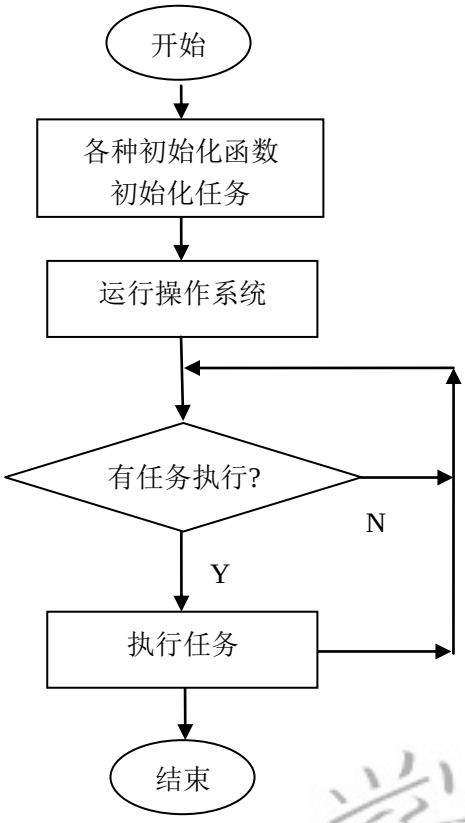

图 3-19 协议栈简要流程

main()函数开始主要是做了一些包括硬件、网络层、任务等的初始化，之后就是最重要的两个函数：初始化操作系统函数 osal_init_system();和运行操作系统 osal_start_system()。main 函数代码如下：

```
int main( void )
{
  osal_int_disable( INTS_ALL );          //关闭所有中断
  HAL_BOARD_INIT();                      //初始化系统时钟
  zmain_vdd_check();                     //检查芯片电压是否正常
  InitBoard( OB_COLD );                  //初始化 I/O ， LED 、Timer 等
  HalDriverInit();                       //初始化芯片各硬件模块
  osal_nv_init( NULL );                  //初始化 Flash 存储器
  ZMacInit();                            //初始化 MAC 层
  zmain_ext_addr();                      //确定 IEEE 64 位地址
  zgInit();                              //初始化非易失变量
```

```
#ifndef NONWK
  afInit();
#endif
  osal_init_system();                          //初始化操作系统
  osal_int_enable( INTS_ALL );                 //使能全部中断
  InitBoard( OB_READY );                       //初始化按键
  zmain_dev_info();                            //显示设备信息
#ifdef LCD_SUPPORTED
  zmain_lcd_init();
#endif
#ifdef WDT_IN_PM1
  WatchDogEnable( WDTIMX );
#endif
  osal_start_system();                         //进入操作系统且不会 back
  return 0;
}
```

第一个重要的函数 osal_init_system()。该函数里面写了 6 个初始化函数包括时钟初始化、电源管理系统初始化、系统任务初始化等，其中关键的是系统任务初始化函数 osalInitTasks()。通过创建在任务表中定义的任务来初始化“任务”系统。系统任务初始化函数 osalInitTasks()用于对 taskID 进行初始化，每完成一个，taskID 加一。

需要用到的硬件的初始化工作在函数 osal_init_system()内最后一个函数 SampleApp_Init(taskID)中完成。本次设计为了减小 Zigbee 芯片的工作开销，Zigbee 模块只进行数据的无线透传，与其他设备间的数据通信采用串口实现。因此，SampleApp_Init(taskID)函数需要先对串口进行初始化 MT_UartInit()，然后登记任务号 MT_UartRegisterTaskID(task_id)。

协议栈中对串口数据处理的函数是 MT_UartProcessZToolData (uint8 port, uint8 event)，它的任务是将收到的串口数据打包、校验后生成一个消息，登记后发给上层。其代码如下：

```
void MT_UartProcessZToolData ( uint8 port, uint8 event )
{
   uint8 flag=0,i,j=0;                    //flag 是收到数据的标识，j 是数据长度
   uint8 buf[128];                        //串口 buffer 最大缓冲默认是 128
   (void)event;
   while (Hal_UART_RxBufLen(port))        //检测串口数据是否接收完成
   { HalUARTRead (port,&buf[j], 1);       //把数据接收放到 buf 中
    j++;                                  //记录字符数
    flag=1;                               //已经从串口接收到信息
    }
    if(flag==1)                           //已经从串口接收到信息
    {
      pMsg = (mtOSALSerialData_t *)osal_msg_allocate( sizeof
```

```
            ( mtOSALSerialData_t )+j+1);
            pMsg->hdr.event = CMD_SERIAL_MSG;
            //事件号用原来的 CMD_SERIAL_MSG
            pMsg->msg = (uint8*)(pMsg+1);
            //把数据定位到结构体数据部分
            pMsg->msg [0]= j;                          //给上层的数据第一个是长度
            for(i=0;i<j;i++)                           //从第二个开始记录数据
            pMsg->msg [i+1]= buf[i];
            osal_msg_send( App_TaskID, (byte *)pMsg );  //登记任务，发往上层
            osal_msg_deallocate ( (uint8 *)pMsg );      //释放内存
        }
    }
```

数据打包并登记任务后，通过 SampleApp_SerialCMD(mtOSALSerialData_t *cmdMsg)函数把数据无线发送出去。其代码如下:

```
    void SampleApp_SerialCMD(mtOSALSerialData_t *cmdMsg)
    {
     uint8 i,len,*str=NULL;                                  //有用数据长度 len
     str=cmdMsg->msg;                                        //指向数据开头
     len=*str;                                  //消息的第一个字节代表数据长度
     if ( AF_DataRequest( &SampleApp_Periodic_DstAddr, &SampleApp_epDesc,
                          SAMPLEAPP_COM_CLUSTERID,
                          len+1,                             //数据长度
                          str,                               //数据内容
                          &SampleApp_TransID,
                          AF_DISCV_ROUTE,
                          AF_DEFAULT_RADIUS ) == afStatus_SUCCESS )
       {
       }else{ // Error occurred in request to send.}
    }
```

当 Zigbee 模块接收到无线数据时，通过函数 void SampleApp_MessageMSGCB(afIncomingMSGPacket_t *pkt)实现。代码如下：

```
    void SampleApp_MessageMSGCB( afIncomingMSGPacket_t *pkt )
    {
     uint8 i,len;
     switch ( pkt->clusterId )
        { case SAMPLEAPP_COM_CLUSTERID:          //如果是串口透传的信息
          len=pkt->cmd.Data[0];
          for(i=0;i<len;i++)
          HalUARTWrite(0,&pkt->cmd.Data[i+1],1);       //发给串口
          HalUARTWrite(0,"\n",1);                       //回车换行
          break;
        }
    }
```

再看第二个重要的函数 osal_start_system()，其功能是系统任务轮询。这个函数会以轮询的方式查询登记发生的事件，若有登记则调用对应的执行函数，若没有则进入低功耗模式。这个函数是永远不会返回的。代码如下：

```
void osal_start_system( void )
{
#if !defined ( ZBIT ) && !defined ( UBIT )
for(;;)    // Forever Loop
#endif
{
uint8 idx = 0;
osalTimeUpdate();
Hal_ProcessPoll();
do {
  if (tasksEvents[idx])
  break;
} while (++idx < tasksCnt);
  if (idx < tasksCnt)
{
  uint16 events;
halIntState_t intState;
HAL_ENTER_CRITICAL_SECTION(intState);     //进入临界区
events = tasksEvents[idx];                //获得待处理的事件
tasksEvents[idx] = 0;                     //清除本次任务的事件
HAL_EXIT_CRITICAL_SECTION(intState);      //退出临界区
events = (tasksArr[idx])( idx, events );   //调取任务处理函数
HAL_ENTER_CRITICAL_SECTION(intState);     //进入临界区
tasksEvents[idx] |= events;               //保存未处理的事件
HAL_EXIT_CRITICAL_SECTION(intState);      //退出临界区
}
#if defined( POWER_SAVING )
else{osal_pwrmgr_powerconserve();         //进入睡眠
  }
#endif
}
}
```

第 4 章 家庭网关设计

家庭网关在智能家居系统中的地位非常关键,它是连接家庭的无线传感网络和 Internet 网络，实现传感器终端与服务器通信的桥梁。家庭网关主要包含三个模块：Zigbee 模块、ARM 模块和网络模块。ARM 模块与 Zigbee 模块通过串口通信，ARM 模块与网络模块通过 SPI 接口通信，网络模块通过 RJ45 接口接入 Internet。由于网关中的 Zigbee 模块是协调器的角色，除了连接无线传感网络和 Internet，网关本身还承担着启动网络、管理网络节点、存储网络节点信息，提供路由消息、安全管理和其它服务的功能。由于第三章已经讲解了 Zigbee 模块，本章主要讲解 ARM 和网络模块，典型应用电路如图 4-1 所示。

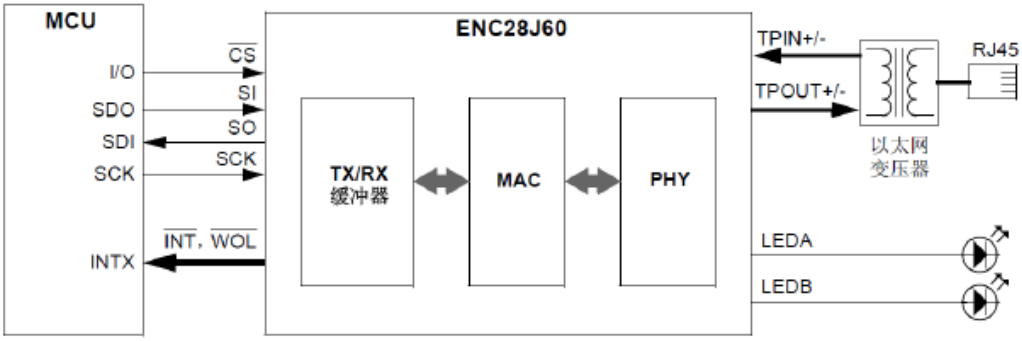

图 4-1 网络模块功能图

4.1 ARM 模块硬件设计

4.1.1 STM32F103 芯片介绍

2006 年 ARM 公司改革了内核设计架构，推出了 ARM7 的后继者——Cortex-M3 内核。其大刀阔斧的改革表现在，首先 Cortex-M3 抛弃了 ARM7 的冯诺依曼架构，改用了 ARMV7 的哈佛架构，使得处理器具有更高的执行效率[35]。其次，Cortex-M3 支持 Thumb-2 指令集，避免了因 Thumb 和 ARM 代码的切换而降低性能。Cortex-M3 创新性地在内核中集成 NVIC，优于使用外部中断控制器的 ARM7。此外，通过使用 tail-chaining 连续中断技术大幅缩短了延迟，改善了响应性能。NVIC 集成的系统节拍计时器不需要外部时钟就可以产生中断时间间隔，同时支持三种睡眠模式的电源管理方案。

除此之外，相对于 ARM7，Cortex-M3 还有许多突出的优势。如更小的基础内核、更快的指令执行速度、更低的功耗、硬件自动完成压栈、单周期乘法指令等等。这些改进显著地简化了编程和调试的复杂度，极大地提高了处理器的性能。

本次设计我们采用 ST 公司的基于 Cortex-M3 内核的 STM32F103ZETT6 芯片作为 ARM 处理器。该芯片配置非常强大，它拥有的资源包括：64KB SRAM、2 个 DMA 控制器（共 12 个通道）、2 个基本定时器、4 个通用定时器、2 个高级定时器、512KB FLASH、3 个 SPI、2 个 IIC、5 个串口、1 个 USB、1 个 CAN、3 个 12 位 ADC、1 个 12 位 DAC、1 个 SDIO 接口、1 个 FSMC 接口以及 112 个通用 IO 口。如此丰富的资源让 STM32F103ZETT6 芯片拥有强悍的性能，再考虑到其较低的零售价格，所以我们选择 STM32F103ZETT6 它作为我们的处理器。

4. 1. 2 复位电路

复位的作用是将系统恢复到起始状态。如图 4-2 所示，电阻 R32 和电容 C51 构成了 STM32 的上电复位电路，同时设置了必要时的手动复位按键。

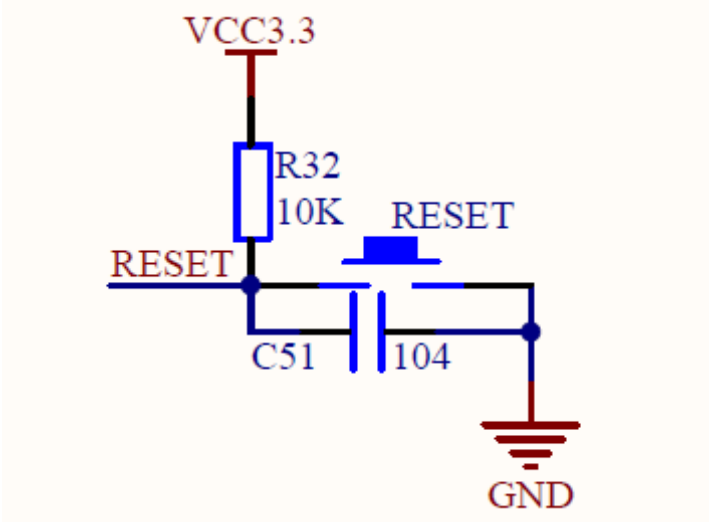

图 4-2 复位电路

4.1.3 JTAG 接口

JTAG 是英文“Joint Test Action Group（联合测试行为组织）”的词头字母的简写。设计 JTAG 接口的作用是对硬件进行仿真、调试，同时也可以用来下载程序。这里我们采用的是标准的 JTAG 接法，如图 4-3 所示。然而，强大的 STM32 不单支持 JTAG 模式，同时兼容只占 2 个 I/O 的 SWD 模式，并且具有更快的下载速度。同时，我们也要使用支持 SWD 模式的仿真器如 JLINKV8、STLink 等。

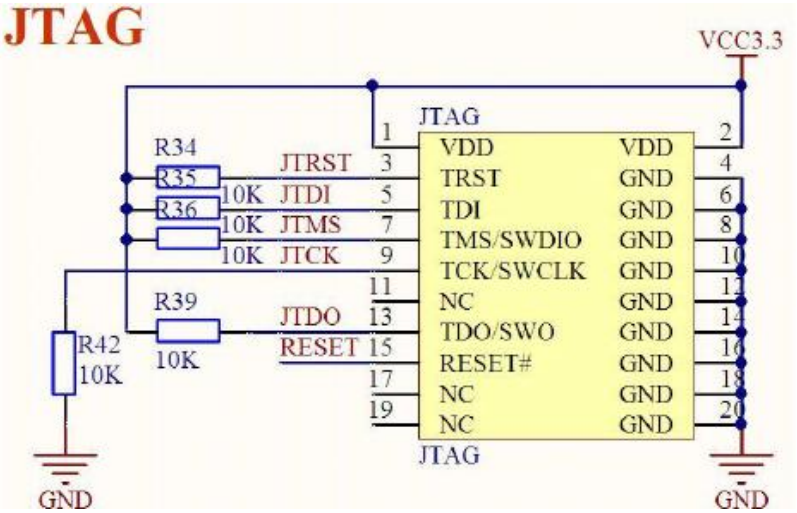

图 4-3 JTAG 接口电路

4.2 以太网模块硬件设计

4.2.1 ENC28J60 芯片介绍

ENC28J60 是带有行业标准串行外设接口（Serial Peripheral Interface，SPI）的独立以太网控制器。ENC28J60 完全支持 IEEE 802.3 规范，通过许多包过滤机制对输入的数据包进行约束。ENC28J60 的使用非常简单，只需要两个中断引脚和一个 SPI 接口，就足以为任何处理器提供以太网接口。其中两个连接 LED 的中断引脚用于指示网络活动状态，SPI 接口可实现高达 10Mb/s 的数据传输速度。除此之外，ENC28J60 内部的 DMA 模块能够实现快速数据吞吐和 IP 校验和计算。

ENC28J60 的主要特点如下[36]：

- 完全遵守IEEE802.3以太网络协议
- 集成 MAC 和 10 BASE-T 物理层
- 拥有全/半双工两种工作模式
- 传输失败时可编程自动重发机制
- SPI 接口速度可达 10Mbps
- 8K 数据接收和发送双端口 RAM
- 支持直接内存存取 DMA 功能
- 可调空间的接收/发送缓冲区
- 两个可编程 LED 输出
- 支持七个中断源

- TTL 电平输入
- 提供多种封装：SOIC/SSOP/SPDIP/QFN 等

4.2.2 ENC28J60 硬件设计

ENC28J60 硬件电路设计如图 4-4 所示。

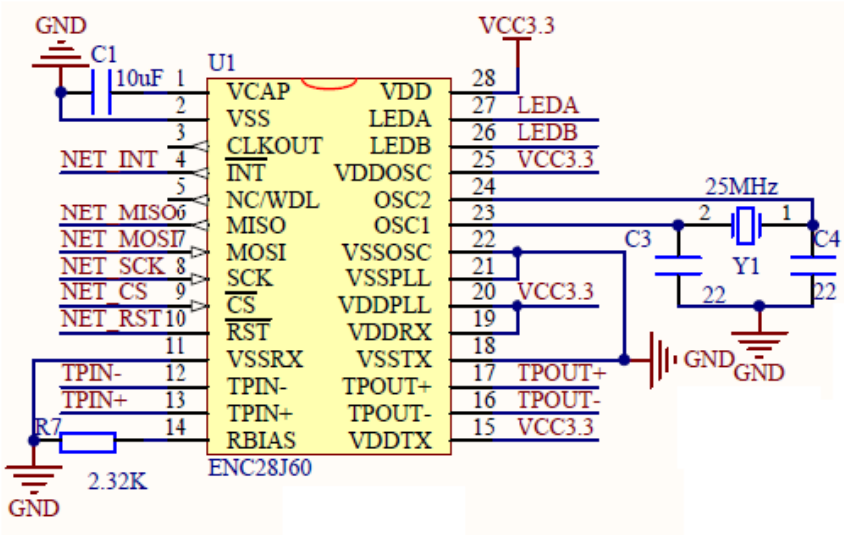

图 4-4 ENC28J60 硬件电路

4.2.3 RJ45 接口设计

RJ45，即 Registered Jack 45，是常用的网络设备互联通信的接口。这种接口俗称水晶头，由八种不同颜色的电芯按一定的次序排列的。其硬件设计如图 4-5 所示。

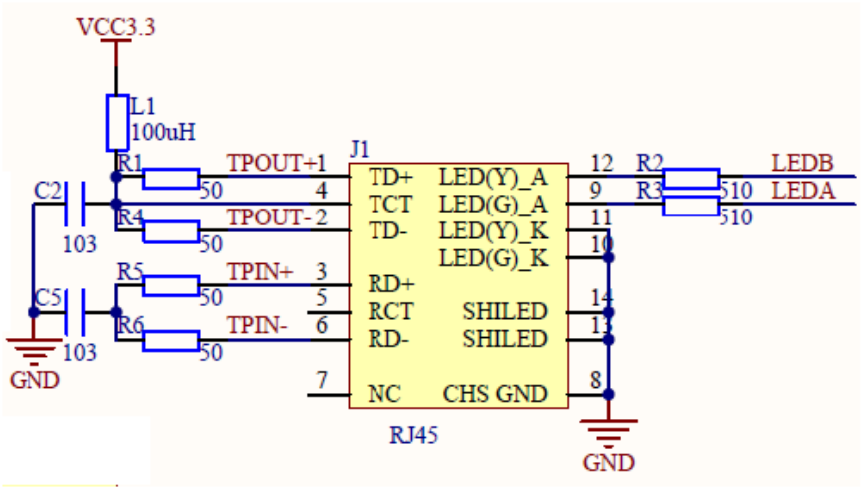

图 4-5 RJ45 接口电路

4.3 uIP1.0 的使用

在使用 uIP 的时候，一般通过如下顺序：

1） 实现应用程序和 uIP 的 UIP_APPCALL 接口函数。

UIP_APPCALL 接口函数是开发者使用 uIP 非常重要的一环，要根据开发者的需要进行各种处理。开发者只需要使用 uIP 提供的那些接口函数如 uip_newdata、uip_acked、uip_closed 等，就可以轻松完成这些处理。另外，如果是 UDP，那么还需要实现 UIP_UDP_APPCALL 回调函数。

2） 初始化网卡并配置好 MAC 地址

3） 初始化 uIP 协议栈。

4） 设置 IP 地址、网关以及掩码

这个和电脑上网差不多，只不过我们这里是通过 uip_ipaddr、uip_sethostaddr、uip_setdraddr 和 uip_setnetmask 等函数实现。

5） 设置监听端口

设置完 IP 地址等信息还需要设置端口号，因为不同的端口对应着不同的服务对象，实现不同的功能。例如我们浏览网页需要用到默认的 80 端口。因此，开发者需要根据自身的需求设置不同的监听端口。总的来说，根据自身的角色可以将端口分为客户端和服务器端两类。当自身用作客户端时，则设置需要远程访问的服务器端口；当自身用作服务器端时，只需监听本地的端口。

6） 处理 uIP 事件

在完成了前面所有的设置之后，就可以将处理 uIP 事件的 uip_polling 函数嵌入开发者的主循环中，实现 uIP 事件的轮询处理。

4.4 家庭网关程序设计

4.4.1 RVMDK3.80A 简介

软件开发 IDE 工具我们选择德国的 KEIL 公司的 RealView MDK，简称 RVMDK。在全球 RVMDK 被超过 10 万的嵌入式开发工程师使用，它集成了业内最领先的技术，包括 μVision3 集成开发环境与 RealView 编译器[37]，如图 4-6 所示。RealView MDK 支持多种嵌入式微处理器，拥有强大的仿真模拟功能和性能分析功能。RVMDK 友好的代码编辑界面和丰富的编辑工具深受开发者的喜爱。

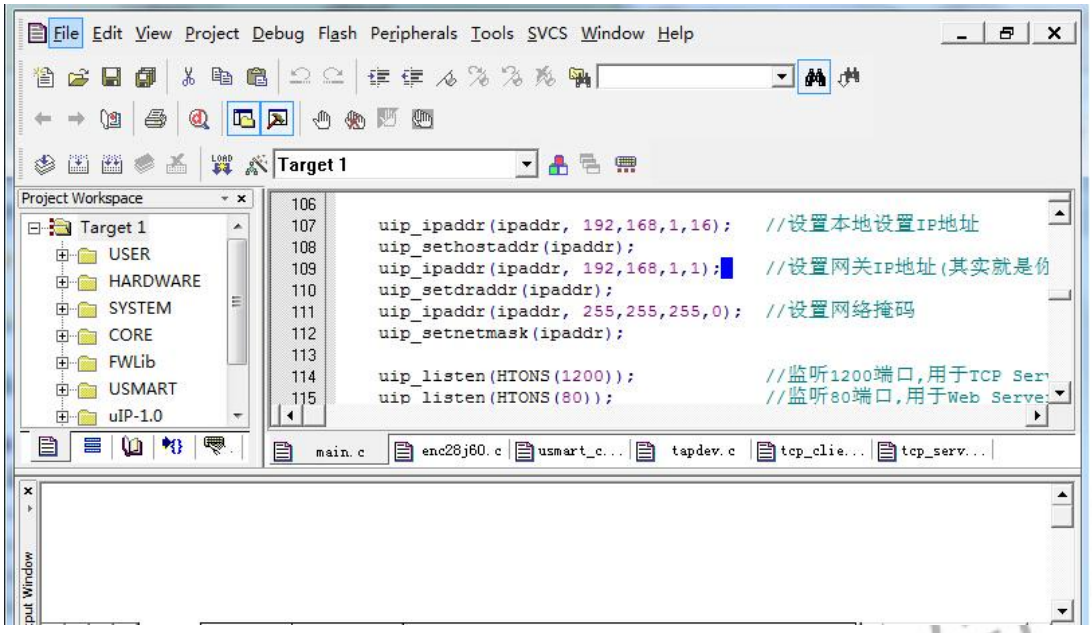

图 4-6 RVMDK3.80A 软件界面

4.4.2 uIP 的移植

实现 uIP 的移植主要是通过 uIP_APP 文件夹中的程序文件，如图 4-7 所示。由 uIP 提供的 clock-arch.c 代码文件的作用是为 uIP 提供时钟节拍，主要是通过该文件中的 clock_time 函数实现。uIP 提供的 tapdev.c 代码文件被用来实现 uIP 与网卡的接口，主要通过该文件中的 tapdev_init、tapdev_read 和 tapdev_send 三个重要函数实现。tcp_demo.c 代码文件主要通过 tcp_demo_appcall 函数实现 UIP_APPCALL 接口，同时通过 uip_log 函数打印日志。tcp_demo_appcall 函数能够依据端口号调用相对应的 appcall 函数，从而实现不同的功能。论文设计的网关是作为客户端去远程访问服务器，实现数据通信要用到 tcp_client_demo.c 文件。

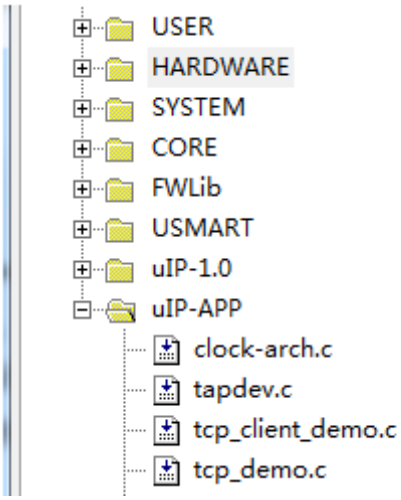

图 4-7 uIP_APP 文件夹

移植第一步：实现在 unix/tapdev.c 里面的三个函数。首先实现用于初始化网卡的 tapdev_init 函数。其次实现用于从网卡读取数据包功能的 tapdev_read 函数。该函数在读取数据后将其存入 uip_buf 缓存区中，再把数据包的字节长度发送给 uip_len。最后还要实现向网卡发送数据包功能的 tapdev_send 函数。在 tapdev_read 函数操作完成后，该函数得把存入 uip_buf 中长度为 uip_len 的数据发送出去。

第二步，由图 2-5 知道 uIP 还需要外部为其提供定时器服务。因此，需要处理器为其提供一个定时器，提供 10ms 的计时服务，通过 clock-arch.c 里面的 clock_time 函数返回给 uIP 使用。

第三步，配置 uip-conf.h 里面的宏定义选项。需要设置的宏定义选项包括：TCP 最大连接数、TCP 监听端口数、CPU 大小端模式等，只需要根据具体开发的需要进行设置。

由此可见，只需完成简单的三个步骤，就可以实现 uIP 的移植。

4.4.3 网关程序设计

论文设计的网关是客户端的角色，也就是实现一个 TCP 客户端的应用。TCP 客户端 appcall 函数——tcp_client_demo_appcall 在 tcp_client_demo.c 里面实现。tcp_client_demo_appcall 函数实现数据收发的方法比较简单。通过 uip_newdata()函数判断是否收到服务器端数据，若收到，则将数据存放在 tcp_client_databuf 缓存区，同时做好接收标记。当需要发送数据时，通过调用 tcp_client_senddata 函数即可。代码如下：

```
void tcp_client_demo_appcall(void)
{
    struct tcp_demo_appstate *s =
     (struct tcp_demo_appstate *)&uip_conn->appstate;
    if(uip_aborted())tcp_client_aborted();                    //连接终止
    if(uip_timedout())tcp_client_timedout();                  //连接超时
    if(uip_closed())tcp_client_closed();                      //连接关闭
    if(uip_connected())tcp_client_connected();                //连接成功
    if(uip_acked())tcp_client_acked();                        //发送的数据成功送达
    if (uip_newdata())                                  //接收到一个新的 TCP 数据包
    {
        if((tcp_client_sta&(1<<6))==0)                        //还未收到数据
        {
            if(uip_len>199)
            {
                ((u8*)uip_appdata)[199]=0;
            }
            strcpy((char*)tcp_client_databuf,uip_appdata);
            tcp_client_sta|=1<<6;                             //表示收到客户端数据
        }
    }else if(tcp_client_sta&(1<<5))                           //有数据需要发送
    {
        s->textptr=tcp_client_databuf;                        //发送数据缓存
        s->textlen=strlen((const char*)tcp_client_databuf);
        tcp_client_sta&=~(1<<5);                              //清除标记
    }
    //当需要重发、新数据到达、数据包送达、连接建立时通知 uip 发送数据
    if(uip_rexmit()||uip_newdata()||uip_acked()||
       uip_connected()||uip_poll())
    tcp_client_senddata();                                    //发送数据给服务端
}
```

main 主函数流程相对比较简单，如图 4-10 所示，先初始化网卡（ENC28J60）和 uIP 等，然后设置本地 IP 地址及监听端口。之后调用 tcp_client_reconnect 函数尝试本地 TCP Client 远程连接外部服务端。若没有连接上，该函数会继续尝试连接服务端，直到连上为止。网关主程序初始化代码如下：

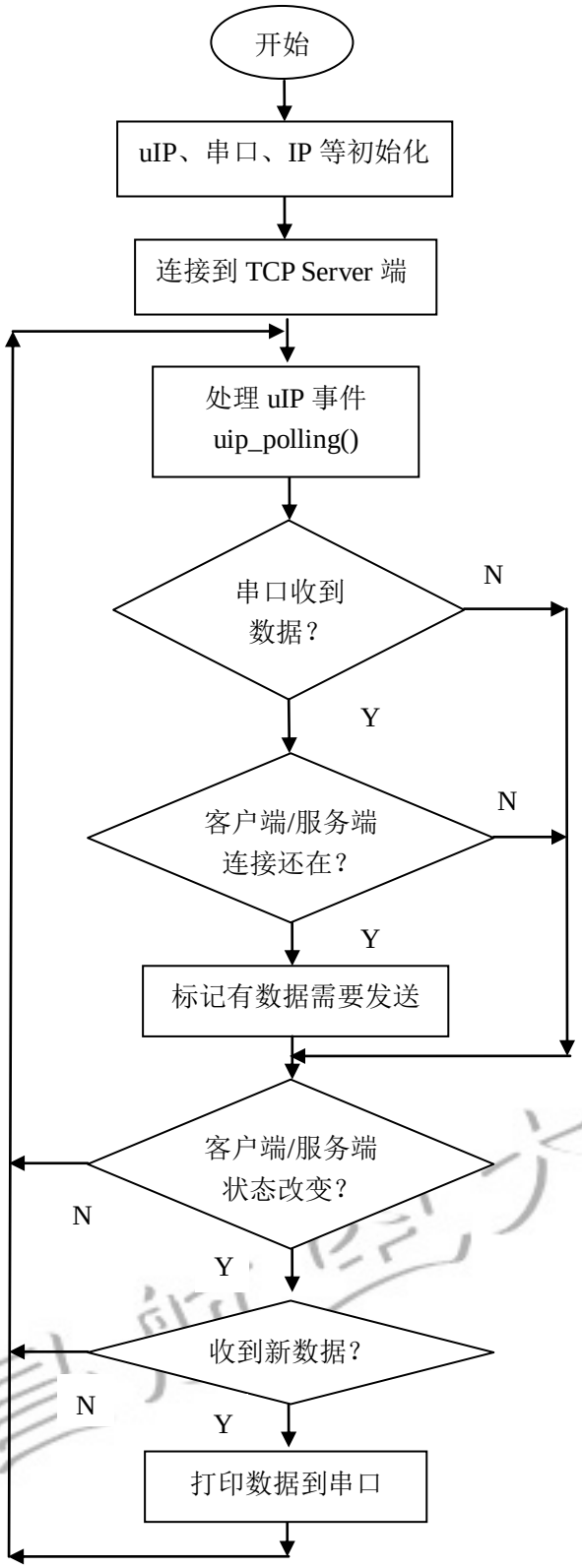

图 4-10 网关主程序流程图

```
int main(void)
{
    u8 tcp_server_tsta=0XFF;
    u8 tcp_client_tsta=0XFF;
    uip_ipaddr_t ipaddr;
    Stm32_Clock_Init(9);                    //系统时钟设置
    uart_init(72,9600);                     //串口初始化为 9600
    delay_init(72);                         //延时初始化
    usmart_dev.init(72);                    //初始化 USMART
    uip_init();                             //uIP 初始化
    uip_ipaddr(ipaddr, 192,168,1,16);       //设置本地设置 IP 地址
    uip_sethostaddr(ipaddr);
    uip_ipaddr(ipaddr, 192,168,1,1);        //设置网关 IP 地址
    uip_setdraddr(ipaddr);
    uip_ipaddr(ipaddr, 255,255,255,0);      //设置网络掩码
    uip_setnetmask(ipaddr);
    uip_listen(HTONS(1200));                //监听 1200 端口,
    uip_listen(HTONS(80));                  //监听 80 端口
    tcp_client_reconnect();                 //尝试连接到 TCP Server 端
```

连接上远程服务端后，便进入主循环代码。最为重要的就是调用 uip_polling 函数开始 uIP 事件的轮询处理。当网关收到串口发来的数据时，先判断客户端或服务端连接是否还在。若在，将数据拷贝到发送缓存区，并标记有数据需要发送。当客户端或服务端的状态发生改变，先判断是否收到了新的数据。若是，则直接将数据打印的串口。网关主程序循环处理代码如下：

```
while (1)
    {
        uip_polling();//处理 uip 事件
        if(USART_RX_STA&0x8000)                 //串口收到数据
        {
            u16 len=0;u16 t=0;
            len=USART_RX_STA&0x3FFF;            //得到此次接收到的数据长度
            for(t=0;t<len;t++)
            {                                   //数据拷贝到发送缓存区
             tcp_server_databuf[t]=USART_RX_BUF[t];
             tcp_client_databuf[t]=USART_RX_BUF[t];
            }
            tcp_client_databuf[t]='\0';         //手动添加结束符'\0'
            if(tcp_server_sta&(1<<7))           //如果连接还存在
                tcp_server_sta|=1<<5;           //标记有数据需要发送
```

```
            if(tcp_client_sta&(1<<7))                    //如果连接还存在
                tcp_client_sta|=1<<5;                    //标记有数据需要发送
            USART_RX_STA=0;
        }   delay_ms(1);
        if(tcp_server_tsta!=tcp_server_sta)//TCP Server 状态改变
        {
            if(tcp_server_sta&(1<<6))                    //收到新数据
            {                                            //打印数据到串口
                printf("TCP Server RX:%s\r\n",tcp_server_databuf);
                tcp_server_sta&=~(1<<6);                 //标记数据已经被处理
            }
            tcp_server_tsta=tcp_server_sta;
        }
        if(tcp_client_tsta!=tcp_client_sta)              //TCP Client 状态改变
        {
            if(tcp_client_sta&(1<<6))                    //收到新数据
            {                                            //打印数据到串口
                printf("TCP Client RX:%s\r\n",tcp_client_databuf);
                tcp_client_sta&=~(1<<6);                 //标记数据已经被处理
            }
            tcp_client_tsta=tcp_client_sta;
        }
    }
}
```

第 5 章 手机终端和服务器端开发

用户通过安装系统配套的手机 APP 便可以控制家居设备，获得良好的交互体验。系统后台服务器端程序采用成熟的技术框架开发，保证功能的稳定性，提供远程控制、数据同步、实时信息显示等服务，使用户可以随时随地掌控家居设备。

5.1 开发语言及环境简介

5.1.1 JAVA 简介

Java 是当今最流行的编程技术之一，是 Sun 公司（2010 年被 Oracle 公司收购）推出的 Java 程序设计语言和 Java 平台（即 JavaSE, JavaEE, JavaME）的总称。Java 技术具有卓越的通用性、高效性、平台移植性和安全性，广泛应用于个人 PC、数据中心、游戏控制台、科学超级计算机、移动电话和互联网，同时拥有全球最大的开发者专业社群[38]。

跨平台特性是 Java 最独到的地方，采用 Java 开发的程序可以轻易部署在 Windows、Solaris、Linux、MacOS 等平台上，节省许多开发流程中的成本。Java 通过虚拟机技术使得采用它开发的程序可以在多平台上运行[39]，而这样的虚拟机是在不同的计算机上虚拟出来的，它的全称为 Java Virtual Machine。通过在运行的平台上模拟实现需要的功能，逐步解释程序，不再关心原始开发平台的特征。Java 虚拟机(JVM)运行程序编译时生成字节码，这种运行过程不与当前的操作系统相关，使得程序可以在多种平台上实现不加修改的移植，符合 SUN 公司提出的口号“Write Once, Run Anywhere（一次编写，处处运行）”。图 5-1 展现了 Java 虚拟机的运行架构，可以清晰的看到它在系统中的位置。

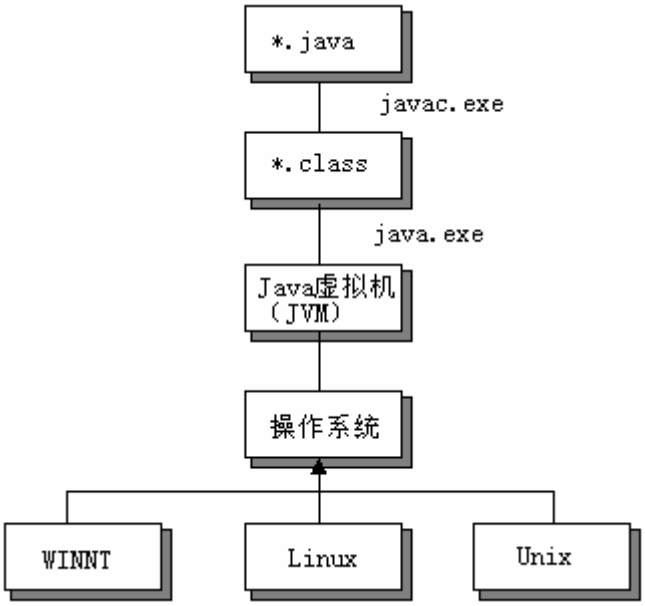

图 5-1 Java 虚拟机运行架构

程序员开发 Java 程序不仅需要在运行平台上搭建虚拟机，还要配置基本的工具包，以得到内置类和其它工具的支持，我们把这样的工具包称为 JDK(Java Development Kit)，它提供了构建 Java 程序、Applet 和组件所必须的开发环境。

5.1.2 Android 简介

Android 是由 Google 公司开发的开放源代码的新一代移动互联网操作系统，遵循了 Linux 的自由开放精神，在平板电脑、智能手机等终端设备上得到了最广泛的应用[40]。据 Gartner 机构统计，目前世界上约有超过二十几亿的移动设备安装了 Android 操作系统。这一装机量直接占据了 49%的操作系统市场，而且随着智能手机的普及，它会得到越来越高的市场份额。

Android 系统是基于 Linux 来开发的，采用被称为软件叠层的方式来构建操作系统、UI、中间件和应用软件几个层次。这样使得每个层次得到很好的分离，每层实现自己的业务逻辑，大大降低了系统的耦合性，上下层的改变影响不到彼此的运行。图 5-2 是 Android 操作系统的软件架构图，从中可以清晰看到四个层次的上下层关系。一般我们把这四个层次概括为：应用程序层、应用程序框架层、系统运行库层和 Linux 内核层，每层有着自己的核心组件[41]。

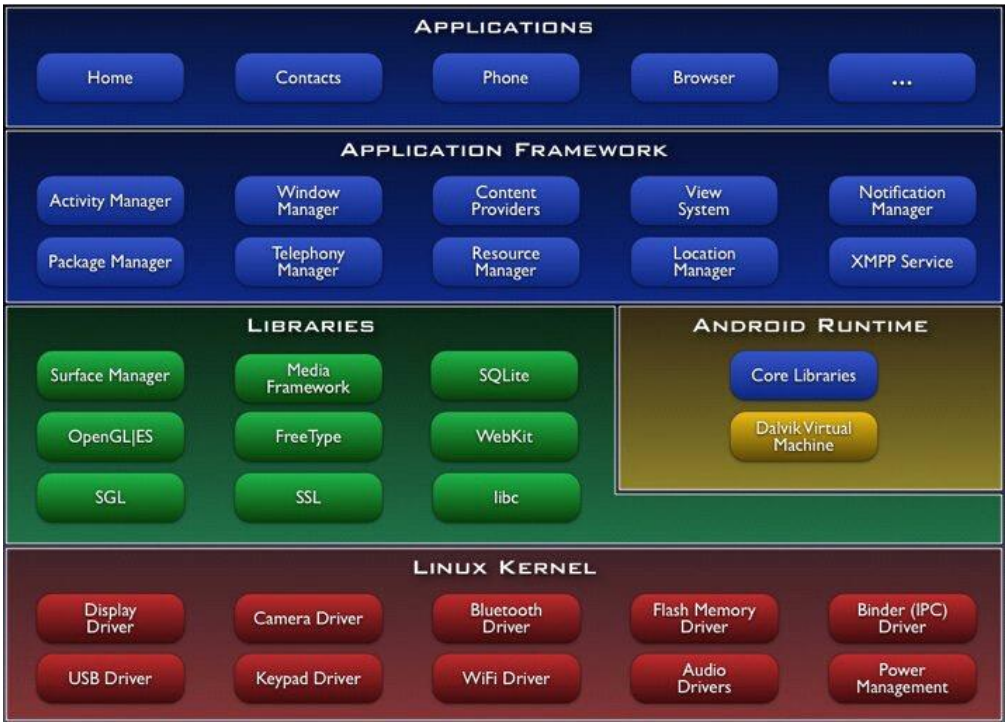

图 5-2 Android 系统架构图

（1）应用程序层

应用程序层提供短信收发、拨号、通讯录、时间日历、地图、Web 浏览器、邮箱等基本的应用，这些应用服务都是最原始的，开发者可以自己定制开发，满足不同用户的需求，这也体现了其开放性。

（2）应用程序框架层

Android 提供给开发者的 API 都封装在应用程序框架层，开发者可以获得所有系统提供的应用服务。也可以在已有的功能模块基础上再重构，创造出更加绚丽的组件，只要遵循 Google 开发者协议，最终呈现给用户的 APP 是安全和丰富的。

（3）系统运行库层

Android 的核心库提供了 Java 开发所需要的基本条件，还包括一些基础的 C/C++库，它们被封装后供需要的组件来调用。为开发者提供服务，创造虚拟运行的环境，是 APP 生命的依靠。

（4）Linux 内核层

Android 是基于 Linux 深度开发，它的最核心的部分还是所采用的 Linux 内核，它来执行内存管理、进程调度、网络通信等操作系统要提供的动作，为硬件和软件架起了沟通的桥梁。

5.1.3 Android 开发环境搭建

（1）首先我们需要从 http://java.sun.com 网站上获得 Java 程序开发所需要的工具包（JDK），安装 Windows/Mac/Linux 环境下需要的 Java 开发工具 JDK。可在命令行输入 java -version 查看已经安装的 Java 版本。

（2）下载 Eclipse。

Eclipse 是一种主流的开发源代码的 Java 开发集成环境 IDE，有众多的插件支持，可以帮助开发高效地进行软件开发。在 Eclipse 网站 (http://www.eclipse.org/downloads/)上有各种平台下的版本供下载。

（3）安装 Eclipse。

解压开下载的 Eclipse 集成开发环境安装包，在安装好 JDK 后，运行 eclipse.exe，设定工作默认目录，便可以进入到开发主界面。

（4）安装 ADT 开发工具。

登录 http://developer.android.com/sdk/installing/installing-adt.html 下载 ADT 插件的最新版本。

（5）下载 Andriod SDK。

SDK 是 Google 提供给开发者的 Android 开发包，里面集成了 Android APP 开发必需的组件，可以从 Android 官方网站（http://www.android.com）或直接访问（http://developer.android.com/sdk/index.html）下载，下载后解压，将解压出来的文件夹放在 eclipse 文件夹中。

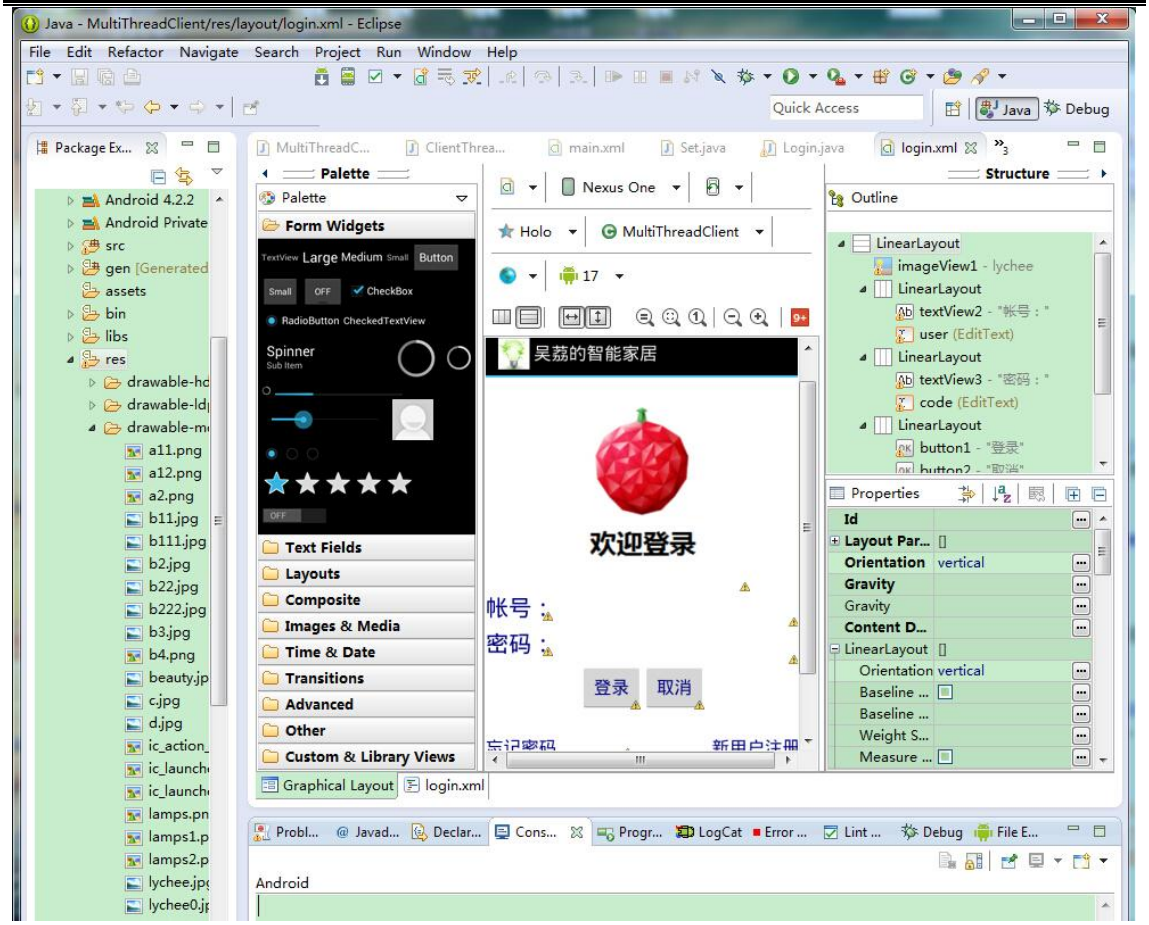

图 5-3 Eclipse 开发环境

图 5-3 是配置好的开发工具界面，里面有不同工作场景，开发者可以按照喜好配置。还可以下载丰富的插件来辅助开发 UI，也可以使用性能调优的一些插件来提高软件的性能。这样的插件在 Eclipse 官网上下载安装，非常方便快捷。

5.2 TCP/IP 协议通信简介

本次设计中，服务端分别与家庭网关和安卓客户端的通信必须遵守网络通信协议。Android 完全支持 JDK 本身的 TCP、UDP 网络通信 API，也可以使用 ServerSocket、Socket 来建立基于 TCP/IP 协议的网络通信；也可以使用 DatagramSocket、Datagrampacket、MulticastSocket 来建立基于 UDP 协议的网络通信。

本次设计我们采用的是基于 TCP/IP 协议的网络通信。TCP/IP 协议是依靠通信的两头对应建立的 socket 对形成网络虚拟链路，从而进行可靠的数据通信。Java 为此提供了非常好的封装，并支持通过 socket 的 I/O 流实现数据通信。

5.2.1 TCP/IP 协议基础

通常所说的 TCP/IP 协议实际上是两个协议的合称，即 TCP 协议和 IP 协议。当不同类型或不同操作系统的计算机要进行通信时，它们必须使用相同的“语言”。IP 协议就是这样一种语言，它可以实现不同计算机间的数据收发功能。但是，IP 协议的缺陷在于它只负责数据的收发，却不考虑数据在传输过程中是否出现了差错[42]。因此，实现了数据通信的收发的同时还需要一种协议可以确保通信的可靠性。TCP 协议就能够很好的解决这个问题。它首先会在通信的两端建立一个通信的虚拟链路，如图 5-4 所示。在发送端发送数据之前，TCP 协议会将数据包按一定的顺序放好，然后在接收端收到数据包后再将其复原。为了确保信息包在传输过程中没有出现差错，TCP 协议采用了重发机制[43]。当发送端发送了一个消息后，若没有收到接收端的确认信息就会重新发送之前的信息。在平常的 Internet 通信中，正是结合了这两种协议才确保了数据在各种复杂情况下的可靠传输。

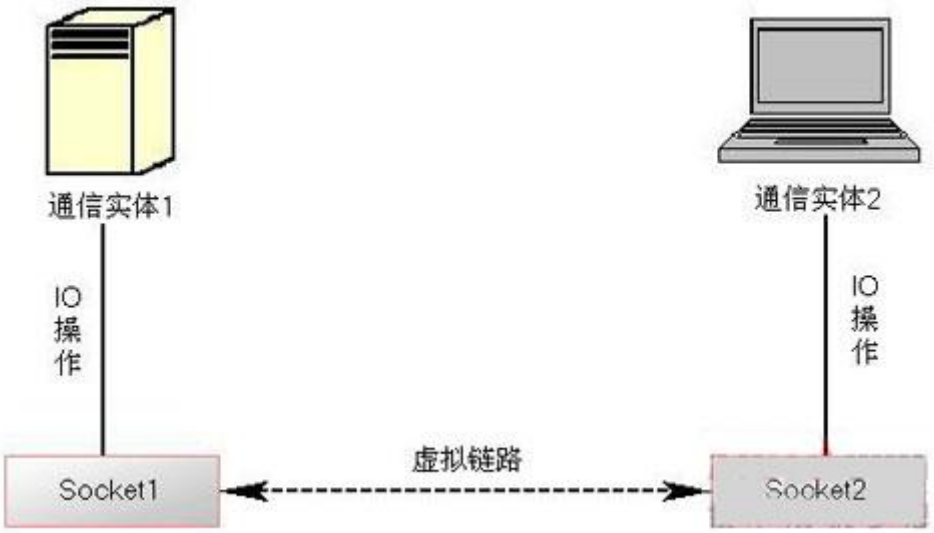

图 5-4 tcp 协议通信示意图

5.2.2 使用 ServerSocket 创建服务器端

正常进行 TCP/IP 通信的两台计算机地位是平等的，没有服务端和客户端的区别。但是在进行通信前，必须先建立 socket 连接，即有一个发出请求且另一个处于等待请求状态。处于等待连接的对象就是服务端，而发出连接请求的就是客户端。

Java 为服务端提供了一个 ServerSocket 类，其中包含了一个用于监听客户端连

接请求的 Socket accept()方法。同时，还提供了几个创建 ServerSocket 对象的构造器。常用的如：SeverSocket(int port):创建一个指定端口的 ServerSocket。

5.2.3 使用 Socket 进行通信

Java 为客户端与指定 IP 地址的服务端建立连接提供了两个 Socket 构造器。

Socket(InetAddress/String remoteAddress,int port):要求知道需要连接的服务端的 IP 及端口信息。

Socket(InetAddress/String remoteAddress,int port,InetAddress localAddr,int localPort): 当本地有多个 IP 信息时，还需要指定是哪一个。

当计算机两端建立了 Socket 连接后，彼此就没有角色的分别了。Socket 为双方提供了 InputStream getInputStream()和 OutputStream getOutputStream()两个方法分别来获取输入输出流。

5.3 安卓手机客户端监控软件开发

Android APP 开发过程采用 MVC 的设计模式，其中的视图是通过使用 XML 文件来布局控制的，而 Java 代码则用来实现业务逻辑。这样用户 UI 从 Java 代码中分离了出来，符合低耦合的开发原则。对于不习惯这种方式的开发者来说，其实可以近似地把 XML 文档看成一个 HTML 页面，它们都通过标记语言来定义用户界面，区别在于使用不同的标签。本节将介绍系统开发的 APP 所实现的功能，通过 APP 的设计来展现家居设备和用户间的交互特点，力求能使控制效果达到最佳，发挥出整套家居系统的性能优势。

5.3.1 客户端 UI 界面开发

Android 应用通常由一个或多个基本组件组成，最常用的组件就是 Activity。它是 Android 应用中负责与用户交互的组件，类似于 Swing 编程中的 Jframe 控件。Android 为 Android 应用提供了可视化用户界面，如果该 Android 应用需要多个用户界面，那么这个 Android 应用将会包含多个 Activity。多个 Activity 组成 Activity 栈，当前活动的 Activity 位于栈顶。

客户端设计需要实现三个基本功能：账户管理、参数配置和指令控制。因此，我们要在 Android 应用的清单文件 AndroidManifest.xml 中定义三个 Activity 来实现

这三个功能，每个 Activity 有不同的视图展现效果，并且在登录界面的 Activity 中设置 <action android:name="android.intent.action.MAIN"/> 和 <category android:name="android.intent.category.LAUNCHER"/>来表明此 activity 是作为应用程序的入口，加载该应用时运行该 Activity。因为本次开发的系统 APP 需要访问网络，需要获得相应的权限，在清单文件中声明该权限<uses-permission android:name="android.permission.INTERNET"/>。

登陆界面如图 5-5 所示，主要是由 5 个线性布局 LinearLayout 构成，每个线性布局之中又布置了不同的组件。首先是占据整个界面的一个外围 LinearLayout,设置其包含的所有子元素的对齐方式为 android:orientation="vertical"垂直排列。在外围 LinearLayout 里面首先是一张“欢迎登录”的图片，向下依次排列的是 4 个子 LinearLayout。每个子 LinearLayout 中又放置了 TextView、EditText、Button 等组件，并设置其排列方式为 android:orientation="horizontal"水平排列。在设置密码输入框 EditText 属性时设置了 android:inputType="numberPassword"，使其成为一个密码框，只接受数字密码，且所有输入的密码会以点号显示以隐藏实际密码。同时，点击时可以发现输入法会自动切换到数字键盘。

图 5-5 登录界面

对于新用户，首先必须点击“新用户注册”从而弹出注册框，输入用户名、密码、邮箱后向服务器发出请求。如果此时服务器端返回正确注册结果，APP 会弹出注册成功消息，用户便可以使用该帐号登录，否则服务器会回滚注册信息，

用户需要重新注册。如果忘记密码，用户可以点击“忘记密码”，此时服务器端会向该用户名的邮箱发送重置后的密码，用户使用该新密码登录。

当用户手机使用的 Wi-Fi 与智能家居网关在同一个网络时，可以登录后直接控制家居设备，因为此时同一个局域网内，网关和 APP 通过使用本地 IP 的方式来建立连接，不要额外的配置。当用户想通过运营商网络来达到远程控制的目的，就需要在首次登录后设置我们为家居设备配备的服务器的 IP 和端口地址。因此，在设计登录界面的时候增加了一个“设置参数”菜单，当点击该菜单时弹出配置界面，如图 5-6 所示。配置界面和登录界面类似，比较简单，也是采用 LinearLayout 布局，点击保存或取消按钮后回到登录界面。

图 5-6 配置界面

用户登录后即进入指令控制的功能界面，如图 5-7 所示。智能家居远程控制通常有三种模式：远程控制设备状态改变、获取当前远程设备状态和与远程设备信息交互。因此，控制界面主要设计了三个功能：远程控制灯的亮灭及调节亮度、获取并显示当前家里温度信息以及在线聊天。控制界面总体是采用 LinearLayout 布局，垂直布置了三个功能的相关组件。

在线聊天功能主要是由一个 EditText 输入框和 Button 按钮完成输入和发送功能，再由一个 TextView 显示聊天记录，TextView 的外面添加了一个 ScrollView 设置垂直滚动条。聊天功能的设计能及时让家人知道互相间的操作，比如温度的调整，这样就避免了重复操作，还能节省能源开支。

控制灯的亮灭是由一个状态开关按钮 ToggleButton 实现的。ToggleButton 是 Button 派生而来，本质上也是 Button，拥有 Button 的各种属性。为了达到美观的效果我们把 ToggleButton 的背景设置为一个灯泡的图片，

android:background="@drawable/toggle_selector"。切换开关时灯泡分别是彩色和黑白的，这是因为这里的灯泡素材是一个制作后的.xml 文件，其代码如下：

```
<selector
xmlns:android="http://schemas.android.com/apk/res/android">
<itemandroid:state_checked="true"
android:drawable="@drawable/bulb_on"
/>
<itemandroid:state_checked="false"
android:drawable="@drawable/bulb_off"
/>
</selector>
```

其中的 bulb_on 和 bulb_off 分别是彩色和黑白色的灯泡图片，当状态开关按钮选中时会选择 bulb_on 图片作为背景，没选中则以 bulb_off 图片作为背景。这样，就实现了灯泡亮灭的效果。灯泡的亮度控制是由一个拖动条 SeekBar 实现的，通过拖动条的滑块位置来标识数值，用户可以拖动滑块改变数值。滑块可以设置其数值的最大值和初始值，另外，滑块的外观也可以设置为自己喜欢的图片背景，如图 5-7 所示。显示温度值比较简单，主要由 TextView 组件实现。

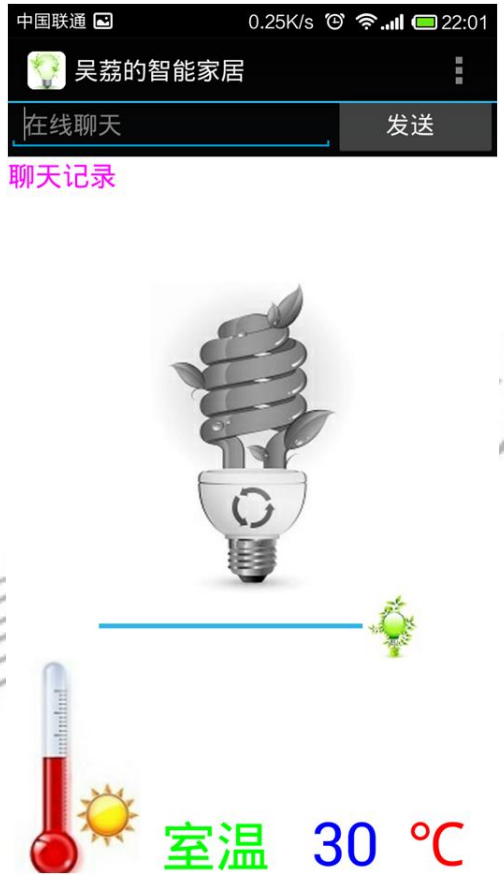

图 5-7 控制界面

5.3.2 客户端控制程序开发

Android 的视图控制文件 XML 生成后，各种资源文件会有一个对应的参数被自动设置，保存在 R.java 文件中。应用程序可以通过 Java 代码 setContentView(R.layout.<资源文件名字>)来调用，接着就可以对该资源进行操纵，改变其在 Activity 中的效果以满足程序设计的需要。XML 的 android:id 属性用来唯一标识使用的对应组件，Java 代码中使用 findViewById(R.id.<android.id 属性值>)来访问需要使用的组件，对该组件的属性、事件等进行操作。

在第一次登录前需要进行用户注册，而在此之前需要在设置页面设置好服务器的 IP 地址和端口号。我们在登录页面设置了一个“设置参数”的菜单，点击即可进入设置页面。程序主要有两步：（1）重写 onCreateOptionsMenu(Menu menu)的方法，在该方法里调用 Menu 对象的方法来添加菜单项；（2）重写 onOptionsItemSelected(MenuItem mi)方法实现菜单项的单击事件响应。关键代码如下：

```
finalintSET = 0x11b; // 定义普通菜单项的标识
    publicboolean onCreateOptionsMenu(Menu menu)
    {   menu.add(0, SET, 0, "设置参数");
        returnsuper.onCreateOptionsMenu(menu);
    }
    @Override
    publicboolean onOptionsItemSelected(MenuItem mi)
    {// 选项菜单的菜单项被单击后的回调方法
        switch (mi.getItemId())//判断单击的是哪个菜单项，并作出响应。
        {caseSET:
            Intent intent1=new Intent(Login.this,Set.class);
            startActivity(intent1);
            break;
        }returntrue;
    }
```

进入设置页面后，就可以输入服务端的 IP 地址信息了，在输入完成后点击“保存”按钮，数据就保存在文件中，点击“取消”就清空输入框。由于需要保存的数据量很小，格式也很简单，我们选择 Android 提供的 SharedPreferences 进行保存。SharedPreferences 非常适合保存 key_value 对类型的配置信息。SharedPreferences 的数据总是保存在 /data/data/<package name>/shared_prefs 目录下，SharedPreferences 数据总是以 XML 格式保存。关键代码如下：

```
SharedPreferences.Editor editor; //为 save 按钮绑定事件监听器
save.setOnClickListener(new OnClickListener()
{  publicvoid onClick(View arg0)
   {
   editor.putString("ip", ip.getText().toString());//存入IP
   editor.putString("port", port.getText().toString());//存入端口
   editor.commit();                              //提交所有存入的数据
                              //获取启动当前Activity的上一个Intent
   Intent intent = new Intent(Set.this,Login.class);//返回登录界面
   startActivity(intent);              // 启动intent对应的Activity
   finish();                           // 结束当前Activity
   }
});
```

应用程序开发使用多线程，有效防止线程阻塞而中断程序运行。一个线程负责指令界面的 Activity 活动，随时将用户的控制数据写入对应的 Socket 输出流；一个线程负责将读取到的对应的 Socket 输入流中的数据显示在视图界面上。同时让ClientThread线程负责网络通信的任务，使用s = new Socket("172.20.15.70",30000)来建立指定 IP 地址和端口的服务器端与客户端的 Socket 连接。这样防止了 UI 线程崩溃，当新线程读取到数据时，通过 Android 提供的 Handler 对象来与 UI 进行信息的传递。具体实现代码如下：

```
new Thread()
{publicvoid run()
   {String content = null;
      Try    //不停读取Socket输入流中的数据。
      {
         while ((content = br.readLine()) != null)
         {
            //读到数据后便发送消息给UI
            Message msg = new Message();
            msg.what = 0x123;
            msg.obj = content;
            handler.sendMessage(msg);
         }  //向UI线程发送消息的Handler对象
      }
      catch (IOException e)
      {
         e.printStackTrace();
      }
   }
}.start();
```

ClientThread 线程还负责将收到的来自 UI 线程发送过来的消息发送给远程服务器，代码如下：

```
revHandler = new Handler()   //接收UI线程的消息的Handler对象
{
   publicvoid handleMessage(Message msg)
   {
      if (msg.what == 0x345) //接收到UI线程中用户输入的数据
      {try                    //将用户数据写入网络
      {os.write((msg.obj.toString()+"\r\n").getBytes("utf-8"));
      }catch (Exception e)
         {e.printStackTrace();
         }
      }
   }
    };
```

子线程收到的消息由 Handler 对象来处理。收到数据时，先要对其进行判断确认是否是温度值数据，如果是的话在文本框来显示，如果不是让它显示在对话框中。代码如下：

```
 handler = new Handler()
{
 public void handleMessage(Message msg)
   {
   if (msg.what == 0x456)                    //判断消息的来源
   {
    data=msg.obj.toString();                 //加上收到的数据
    if(data.regionMatches(0, tHead, 0, 3)) //判断数据是否是温度值
    {                                  //去掉温度数据的头，并显示温度值
       temp.setText(data.substring(3));
    }
    elseshow.append(data+"\n");          //将聊天内容显示在对话框
    }
    }
};
```

在线聊天功能是用户在输入框输入后，点击发送按钮将信息传递给服务器。这里要为发送按钮的单击事件添加事件监听器，当用户按下发送按钮后，将 input 输入框中用户输入的数据封装成 Message，然后发送给子线程的 Handler。代码如下：

```
send.setOnClickListener(new OnClickListener()
```

```
{
    publicvoid onClick(View v)
    {
        try
        {
        Message msg = new Message();
        msg.what = 0x345;
        msg.obj =getPreference("user")+":
        "+input.getText().toString();
        clientThread.revHandler.sendMessage(msg);
        input.setText("");                          // 清空input文本框
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
    });
```

对灯泡的状态开关按钮 ToggleButton 添加 OnCheckedChangeListener 监听器，对拖动条 SeekBar 添加 OnSeekBarChangeListener 监听器，使程序能够响应状态或位置的改变，并将改变的状态信息封装成 Message，然后发送给子线程的 Handler，方法与发送聊天信息类似。

5.4 服务器端功能介绍及程序开发

5.4.1 服务器端功能介绍

由于 Ipv4 地址资源紧张，家庭宽带的 IP 地址通常都是宽带运营商分配的，并不是固定的。而手机等终端设备无线上网的 IP 地址通常也是移动通讯运营商动态分配的，也不是固定的。这就使得手机终端不能直接远程控制智能家居设备，需要一个拥有公网 IP 地址的服务器。拥有固定 IP 地址的服务器可以让智能家居网关和手机访问到自己，从而实现数据的转发与处理。

服务器端考虑到了大用户量的需求，采用多线程编程，提升程序的并发性能，可以同时响应众多用户的请求，不会出现阻塞的情况。当用户注册账户时，服务器端要先对用户输入的用户名和邮箱进行检测，在 MySQL 数据库中查询，判断该用户是否注册。如果已经注册，则返回字符串“用户已存在，请重新注册”；如果

确认是新用户则存储该帐号的信息，而且先要对密码进行加密，数据库中存入密码的密文，防止泄漏造成的安全隐患。用户点击“忘记密码”操作时向服务器请求重置密码，服务器端接收到请求先判断该账户信息，通过 JavaMail 来实现邮件的代理。如果正确则向其注册邮箱发送随机的重置密码。用户每一次登录都要向服务器端进行认证请求，等待服务器端返回的查询结果。

当用户使用 APP 连接服务器后，服务器端程序随即开启了一个线程来保障该用户的通信服务，监听每一次请求，并且把使用统一账户登录的人分为一组，如果该账户使用在线聊天功能，则把收到的信息，推送给该组内当前登陆的的客户端，这是通过轮询 socklist 来实现的。

图 5-8 是每一个线程的工作流程图，客户端与服务器建立连接后不断监听数据，收到数据后对请求的类型做判断并返回应答结果，而此时家居系统 APP 将接收到的数据通过 Handler 对象以消息的形式传递给 UI，呈现在用户的眼前。

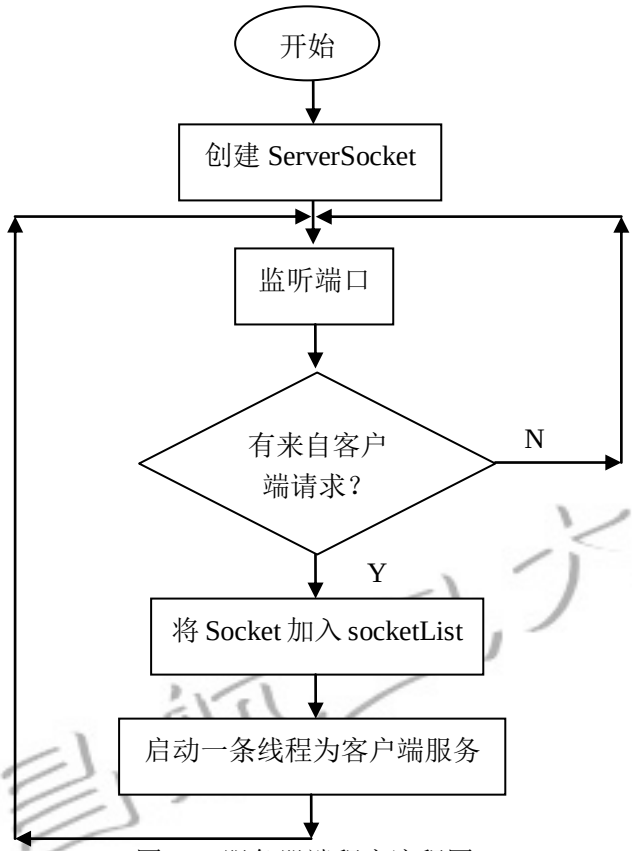

图 5-8 服务器端程序流程图

5.4.2 服务器端程序设计

与服务器端建立连接后，不同类型的数据请求会在代码主体里面做对应的操作，如接收到需要存储在数据库中的数据时，在数据库连接池中取一个连接对数

据库进行操作，如果这一过程中出现阻塞、资源竞争，就需要在向客户端返回错误信息的同时进行回滚操作。服务器监听主类程序如下：

```
public class MyServer
{
   public static ArrayList<Socket> socketList
      = new ArrayList<Socket>();
    public static void main(String[] args)
      throws IOException
   {   //创建一个ServerSocket，用于监听客户端Socket的连接请求
      ServerSocket ss = new ServerSocket(30000);
      while(true)              //采用循环不断接受来自客户端的请求
      {
       //每当接受到客户端Socket的请求，服务器端也对应产生一个Socket
         Socket s = ss.accept();
         socketList.add(s);    //将socket加入socketList集合中保存
         new Thread(new ServerThread(s)).start();
      }//为该socket启动一条线程
    }
}
```

另外，服务器还有负责处理每个线程通信的线程类，如在线聊天功能的数据处理：

```
public void run()
   {
      try
      {       //采用循环不断从Socket中读取客户端发送过来的数据
         String content = null;
         while ((content = readFromClient()) != null)
         {   //遍历socketList中的每个Socket，
            for (Socket s : MyServer.socketList)
            {
               OutputStream os = s.getOutputStream();
               os.write((content + "\n").getBytes("utf-8"));
            }
         }
       }
      catch (IOException e)
      {
         e.printStackTrace();
      }
   }
```

第 6 章 系统运行与测试

前面各章在讲解了系统的各个部分的软硬件设计之后，我们有必要对系统的各个模块进行单独的性能测试，以检测设计是否达到预期目标。之后再将各个模块整合起来，测试整个系统的性能。

6.1 Zigbee 通信功能测试

为了测试我们的 Zigbee 节点的硬件设计是否合格，我们设计了一个串口透传的测试程序。其功能是将 Zigbee 节点通过串口连接到电脑，再通过串口调试助手收发数据，以检测 Zigbee 模块相互通信功能是否正常。我们准备了 4 个 Zigbee 节点，1 个协调器节点，2 个路由器节点和 1 个终端节点，组成一个完备的 mesh 网络，测试示意图如图 6-1 所示，实物图如图 6-2 所示。

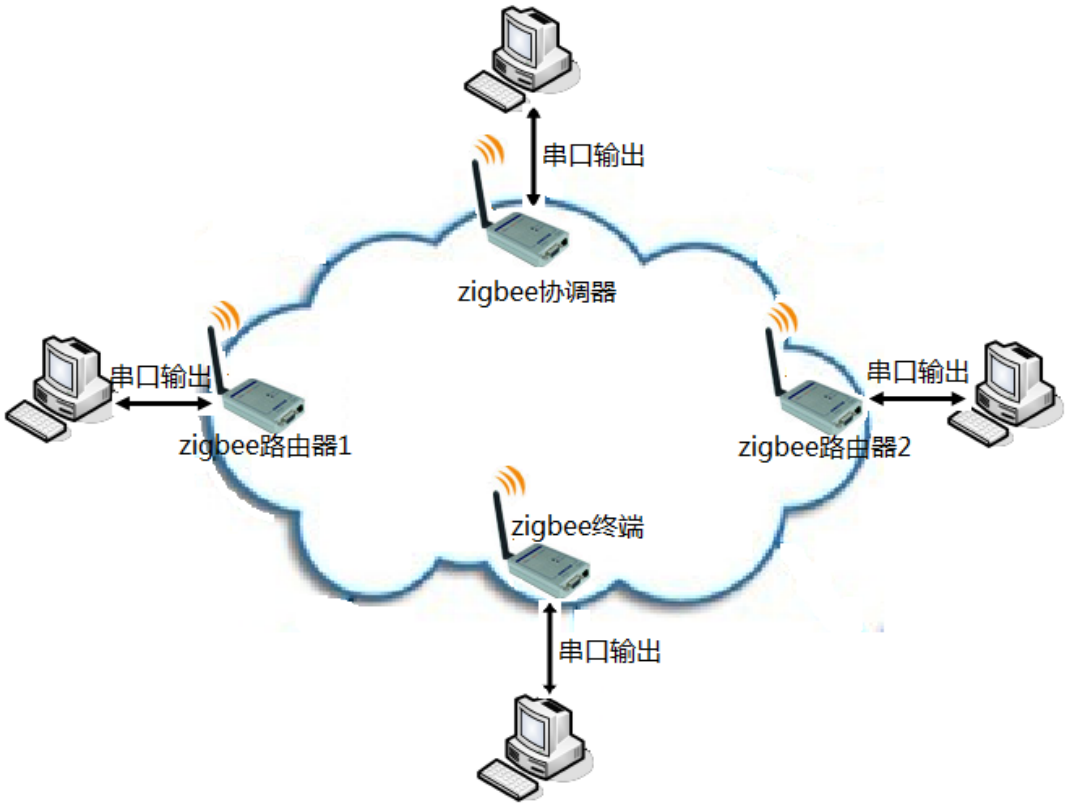

图 6-1 Zigbee 透传测试示意图

图 6-2 Zigbee 透传测试实物图

在 4 个 Zigbee 节点构成的通信网络中，每个节点可以通过计算机串口发送无线数据，也可以将收到无线数据通过串口返回到计算机进行查看。如图 6-3 所示，计算机的串口 COM2、COM9、COM10、COM37 分别连接到协调器、路由器 1、路由器 2、终端，在每个串口的发送窗口都写上了自己的角色名字，如：I am Coordinator。依次点击每个串口助手的发送按钮，发现每个串口都收到了其他节点发送的数据，证明了 Zigbee 软硬件设计的合理。

图 6-3 Zigbee 透传测试数据

6.2 Zigbee 节点无线传输质量检测

在测试了 Zigbee 通信功能达到设计目标后，对于其通信的质量如何，特别是误包率，因此设计了如下测试。我们准备了两个 Zigbee 节点，一个作为发射节点，循环不断的发送数据包；另一个作为接收节点，不断地接收数据包，并将数据包的序数、误包率及之前 32 个数据包的 RSSI 平均值通过串口打印在计算机上，如图 6-4 所示。由于两个节点距离比较近且无遮挡，误包率 PER 很低，但当增大节点距离时，误包率明显上升，如图 6-5 所示。

图 6-4 Zigbee 无线传输质量测试

从图 6-5 曲线图可以看出，当节点通信距离在 15 米以内时，无论是室内有障碍环境还是室外空旷环境，通信丢包率都较低。当通信距离大于 15 米后，室内丢包率开始显著升高，而室外丢包率在较远距离内始终保持较低的丢包率。结合家庭住宅环境的特点，此数据表明 Zigbee 技术比较适用于家庭环境。

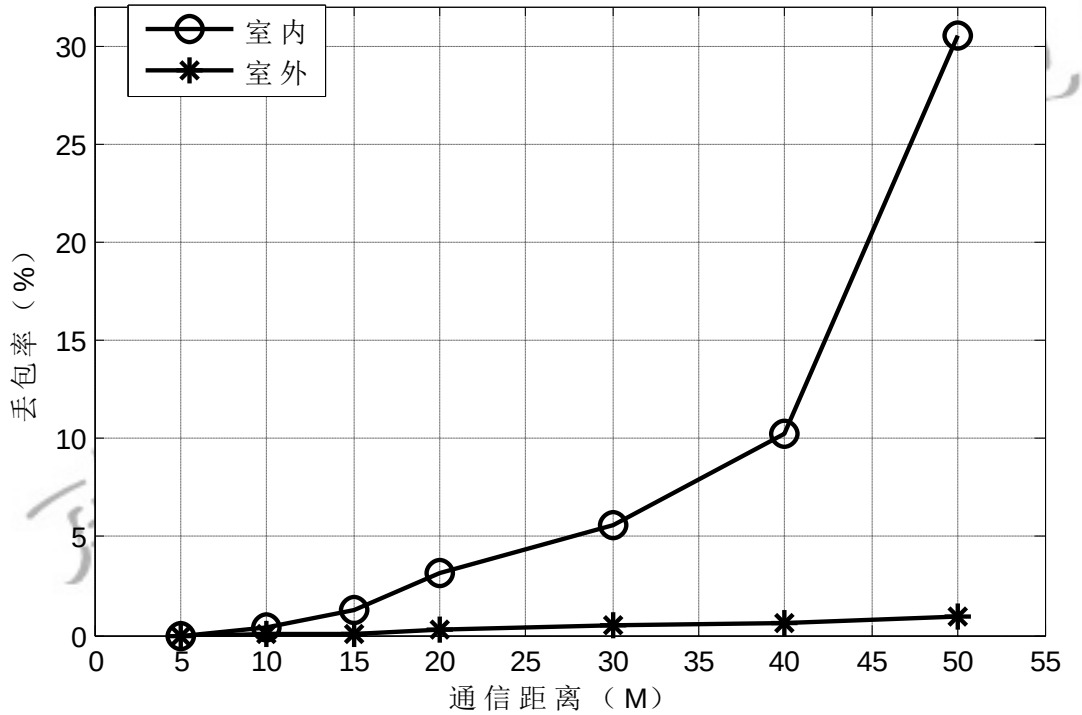

图 6-5 Zigbee 无线通信距离测试

6.3 网关性能测试

论文设计的网关是作为客户端主动去连接服务端，为了测试网关设计能否达到预期性能，我们进行了如下测试。如图 6-6 所示，我们将网关和计算机通过路由器连接到局域网，再将网关的串口通过串口模块连接到计算机串口。

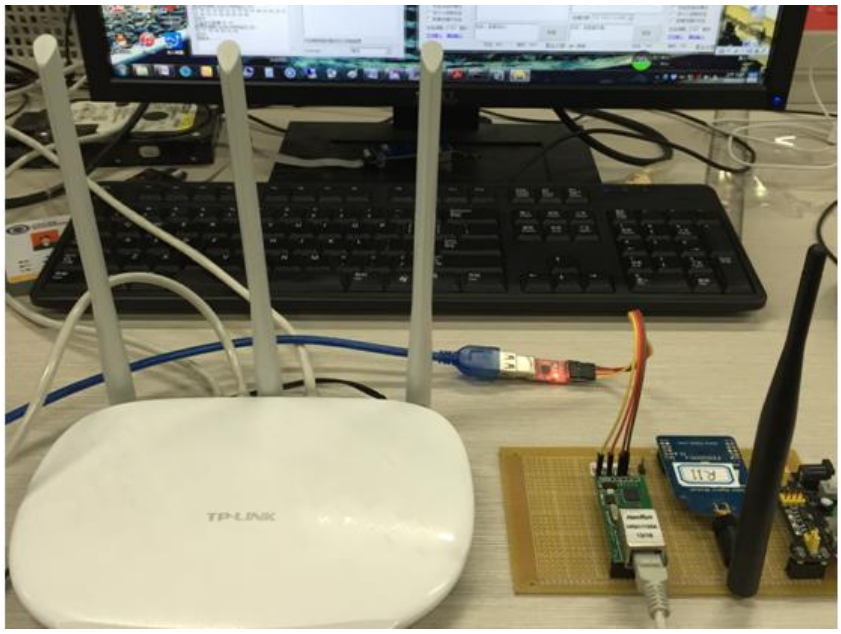

图 6-6 Zigbee 无线传输质量检测数据

这样，我们实现了以计算机做服务器，网关做客户端，并通过串口转网络调试助手查看服务器和网关间的数据通信。但在测试之前要对网关进行配置，主要是对 IP 地址和端口、波特率等的设置，如图 6-7 所示。

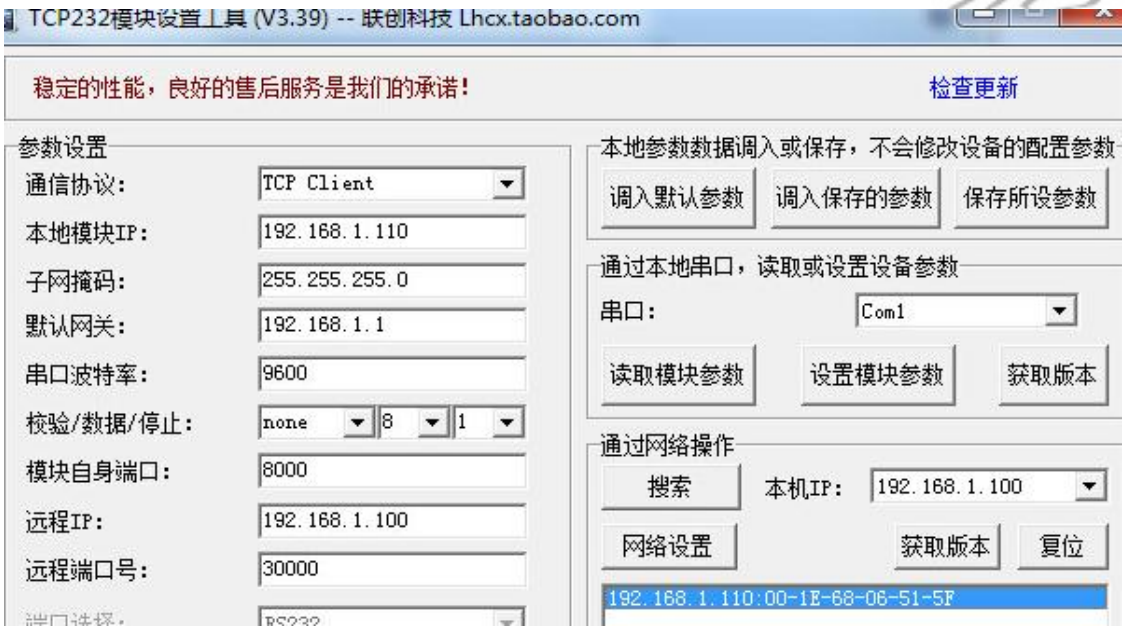

图 6-7 对网关模块的参数设置

我们通过 TCP232 模块设置工具对网关以及服务器端的 IP 地址和端口、默认网关、波特率进行设置。通信协议设置为 TCP Client，默认网关与远程 IP 等信息可查询本地计算机网络连接详细信息，模块的 IP 地址可以设置为局域网中任意不冲突的空闲 IP。保存后重新上电。由于网关和计算机在局域网里，我们在计算机 DOS 窗口尝试 PING 网关 IP。如图 6-8 所示，PING 测试结果显示计算机能够 PING 通网关 IP，表明网关的 IP 设置成功，网络连接正常。

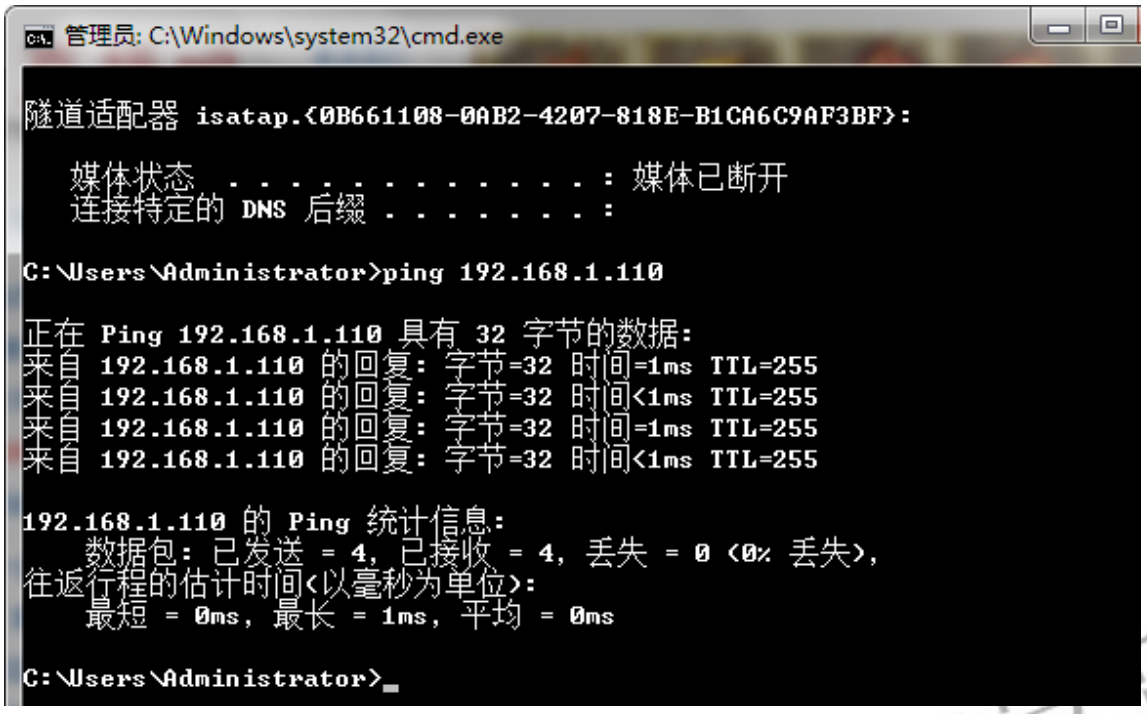

图 6-8 PING 网关 IP 图

然后我们打开串口转网络调试助手，如图 6-9 所示，设置好左边的串口设置和右边的网络设置，注意协议类型选择 TCP Server。分别点击串口打开和开始监听按钮，可以看到按钮的灯亮了，同时在网络这边的连接对象处显示出我们设置的网关的 IP 地址和端口，说明我们的网关已经访问到计算机服务器。我们在串口输入框输入“你好，我是网关”，点击发送，可以看到信息显示在了网络数据接收串口，说明服务器收到了网关发送的数据。然后在网络模块输入框输入“你好，我是服务器”，点击发送，可以看到信息显示在了左边的串口数据接收窗口，说明网关也可以收到服务器端发送的数据。

图 6-9 网关与服务器通信

6.4 安卓软件测试

为了验证我们设计的安卓软件能否通过 Socket 与服务器进行数据通信，我们做了下面的测试。首先，打开网络调试助手，设置网络设置后点击“开始监听”。然后，我们连上手机 Wi-Fi，使手机和计算机连在一个局域网。打开手机安卓软件后配置好服务器 IP 和端口，点击登录，可以看到网络调试助手显示了连接对象的 IP 地址和端口。正是我们手机的 IP 地址，说明手机软件与计算机服务端建立了 socket 连接。我们点击手机上控制灯泡亮灭的图标，灯亮和灭的时候网络助手分别收到数据_L_200 和_L_0，其中_L_是数据的头表示是灯的数据，后面的数值是亮度值。我们滑动滑块后，网络助手收到数据_L_119。我们再在网络助手的输入框中输入 Hello World，点击发送，安卓软件的通话记录显示出收到的数据如图 6-10 所示。表明我们的安卓软件能够和服务器进行预期的数据通信功能。

图 6-10 安卓软件与服务器通信

6.5 系统整体测试

对系统的三大部分进行独立测试之后，我们对整个系统进行测试。我们打开已经安装在手机上的该软件，进入控制界面后点击灯泡按钮，图标变成彩色，同时看到嵌入了 Zigbee 模块的灯点亮，如图 6-11 所示。移动灯泡按钮下面的滑块，可看到灯的亮度相应发生改变。

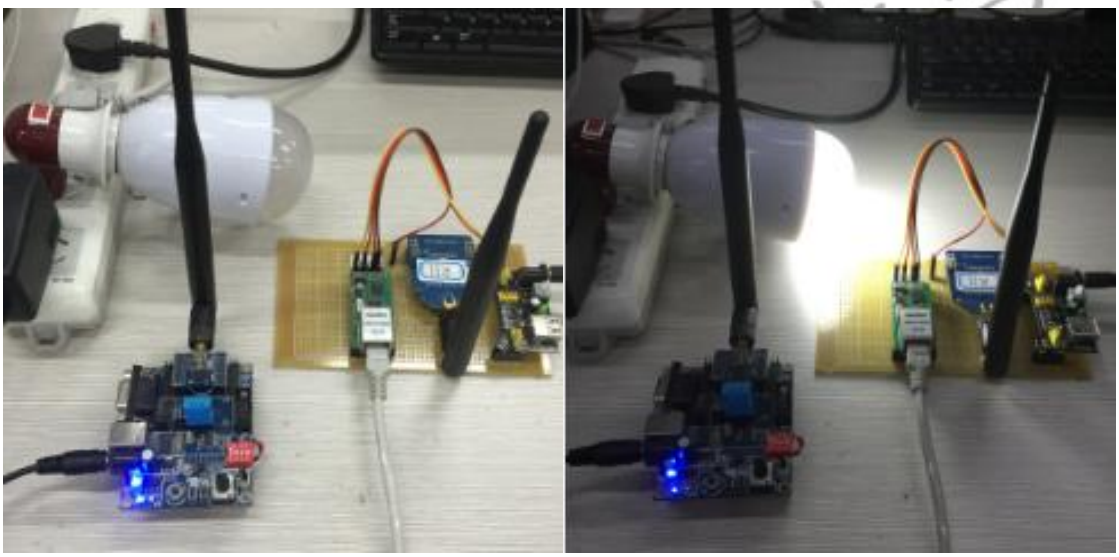

图 6-11 系统整体测试图

我们再使用一部安卓手机并安装上该软件，完成相关设置操作之后相互发送信息。如图 6-12 所示，我们看到了两部手机的聊天记录。同时，在页面底部均显示了当前的温度信息，且当我们用手触摸 Zigbee 底板上的 DHT11 传感器时，发现而且手机端的温度值能够实时更新。

图 6-12 安卓软件与服务器通信

第 7 章 总结与展望

7.1 总结

论文对智能家居的发展概况及国内外发展现状进行分析后，提出了一种基于 Zigbee 的智能家居系统设计方案。文中对 Zigbee 核心板、底板硬件电路以及传感器设备硬件电路进行了设计，同时通过 Z-stack 协议栈对 Zigbee 组网通信程序进行了设计。论文还对负责将家庭传感网络与计算机网络相连的家庭网关硬件及 uIP 协议栈的移植进行了设计。之后，文章设计了基于安卓系统的监控软件和实现远程控制的服务端程序。最后，依托深圳先进技术研究院机器人重点实验室的条件完成了相关软硬件开发与调试，并对系统的各个模块及整体进行了调试，测试结果显示各项设计指标基本实现，证明了系统设计的合理性。

7.2 展望

虽然从最后的测试结果来看，论文实现了预期的设计目标。但是由于时间的关系，本次设计只根据智能家居的三种控制模式：控制设备、获取信息和信息交互，分别设计了灯的亮度调节、温度信息的读取和在线聊天三个功能。在后续的工作中，本人计划设计更多的传感设备如红外入侵、烟雾传感器等，实现更多更丰富的智能家居体验。另外，在安卓控制软件的设计上进一步优化，以实现更加美观的界面和更加便利的操作。

参 考 文 献

[1] 薛磊. NeST 系统智能家居产品交互设计研究[D]. 北京交通大学, 2013.
[2] 于春娣. 基于无线传感器网络的目标跟踪技术研究[D]. 南京航空航天大学, 2013.
[3] 杨斌, 李波. 物联天下智能家居[J]. 信息系统工程, 2013 (12): 31-35.
[4] 朱小丽. 基于 CAN 总线的智能家居控制系统[D]. 合肥工业大学, 2012.
[5] 金星. 嵌入式智能家居系统的总体设计与实现[D]. 江西农业大学, 2013.
[6] 何遥. 智能家居发展瓶颈突破在即?[J]. 中国公共安全, 2014 (8): 48-51.
[7] 宋滟泓. 斥资 32 亿美元谷歌购得智能家居游戏入场券[J]. IT 时代周刊, 2014, 1.
[8] 周平春. 浅谈智能家居发展趋势分析[J]. 城市建设理论研究 (电子版), 2014 (2).
[9] 宋滟泓. 智能家居概念急剧蹿红互联网企业开打新战役[J]. IT 时代周刊, 2014 (9): 40-41.
[10] 芦潇静. 智能家居: 由天马行空到万马奔腾[J]. 单片机与嵌入式系统应用, 2014, 14(3): 82-83.
[11] 李佳师. 物联网: 起个大早不能赶个晚集[J]. 计算机光盘软件与应用, 2014, 17(5): 45-46.
[12] 杨祖泽. 浅谈国内智能家居市场的发展和推广[J]. 中国公共安全, 2013 (13): 162-164.
[13] 文竹. Wi-Fi 在定位领域的应用[J]. 数字通信世界, 2011 (12): 24-25.
[14] Lee J S, Su Y W, Shen C C. A comparative study of wireless protocols: Bluetooth, UWB, Zigbee, and Wi-Fi[C]//Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE. IEEE, 2007: 46-51.
[15] Baronti P, Pillai P, Chook V W C, et al. Wireless sensor networks: A survey on the state of the art and the 802.15. 4 and Zigbee standards[J]. Computer communications, 2007, 30(7): 1655-1695.
[16] Baker N. Zigbee and Bluetooth: Strengths and weaknesses for industrial applications[J]. Computing and Control Engineering, 2005, 16(2): 20-25.
[17] Gill K, Yang S H, Yao F, et al. A Zigbee-based home automation system[J]. Consumer Electronics, IEEE Transactions on, 2009, 55(2): 422-430.
[18] 董珍珍, 杨云. 浅析 Zigbee 技术及其应用[J]. 科技视界, 2012 (24): 209-210.
[19] 杨桂松, 王中杰, 何杏宇. 无线传感器网络单跳扩展增强树型路由协议研究[J]. 计算机科学, 2012, 38(12): 88-91.
[20] 王珏. 基于 Zigbee 技术的健康监测系统设计与研究[D]. 武汉理工大学, 2009.
[21] 姚卫国. 嵌入式处理器 MIPS 和 ARM[J]. 环球市场信息导报 (理论), 2012 (2): 50-50.
[22] 李家武, 张敏, 张芳. uIP 协议栈在 DSP 声信号采集阵列上的应用[J]. 现代应用物理, 2013 (4): 379-383.
[23] 任强, 孙玉国. 基于 Web 和蓝牙的智能家居控制系统设计[J]. 信息技术, 2013, 37(10): 39-42.
[24] 彭洪柱. 蜜蜂传递蜜源的信息是蜜蜂的 “8” 字舞吗?[J]. 蜜蜂杂志, 2012, 32(8): 12-12.

[25] Tian L G, Li M, Chen Z L, et al. Design of smart home control terminal based on Zigbee and electronic technology[M]//Advances in Mechanical and Electronic Engineering. Springer Berlin Heidelberg, 2012: 339-344.

[26] Kinney P. Zigbee technology: Wireless control that simply works[C]//Communications design conference. 2003, 2: 1-7.

[27] 李巧英, 刘朝晖. 无线传感器应用在河流防洪预警方面的研究[J]. 电脑知识与技术: 学术交流, 2013, 9(8): 5021-5024.

[28] 张兰. 浅谈 Zigbee 无线通信技术特点及应用[J]. 计算机光盘软件与应用, 2014, 17(1): 297-297.

[29] 宋志月. Zigbee 无线传感器网络在瓦斯监测系统中的应用[J]. 软件, 2011, 32(2): 46-48.

[30] 郝捷. Zigbee 无线传感器系统的设计实现与 LTCC 技术研究[D]. 西安电子科技大学, 2009.

[31] 王财宝. 基于 Zigbee 技术的输电线路导线接头温度在线监测[D]. 上海交通大学, 2010.

[32] 陈鑫洋. 基于 PWM 技术的低压大功率高效率电源的设计[J]. 电子制作, 2014, 8: 013.

[33] 田芳明, 杨丽茹, 金松海, 等. 基于 PIC 单片机的分布式无线温湿度采集系统[J]. 黑龙江八一农垦大学学报, 2011, 23(1): 79-82.

[34] Alliance Z B. Technical specifications-Zigbee 2007 specification[J].

[35] 王哲宇, 朱诗倩, 刘锦高. 嵌入式引导加载程序的可裁剪性分析[J]. 信息技术, 2013 (7): 106-109.

[36] 邹冬华. 基于以太网的嵌入式电表的设计[D]. 南昌大学, 2009.

[37] 刘波. 基于嵌入式 ARM 系统的应用分析[J]. 商场现代化, 2010 (11): 20-20.

[38] 徐静. Java 编程的两条技术路线[J]. 黑龙江气象, 2013, 30(3): 38-38.

[39] 邓洋春. Java 虚拟机关键机制研究与实践[D]. 中南大学, 2009.

[40] 周雯. Android 广播监听机制的两种实现方法[J]. 现代计算机: 上下旬, 2013 (11): 48-49.

[41] 包依勤. 基于智能终端应用的计算机专业课程建设探讨[J]. 物联网技术, 2012, 2(7): 66-68.

[42] 王利矛. 浅谈 OSI/RM 在 TCP/IP 网络中的实现[J]. 福建电脑, 2009 (2): 55-55.

[43] 陈派樟. 基于 TCP/IP 协议的常见攻击方法[J]. 宁德师专学报: 自然科学版, 2010, 22(1): 56-58.

攻读硕士期间发表论文和科研情况

[1] Wu L, Wang L, Su S, et al. Wireless sensor network based solution for nuclear radiation detection[C]//Information Science and Technology (ICIST), 2014 4th IEEE International Conference on. IEEE, 2014: 397-400.

[2] Wang L, Wu L, Wei J, et al. Web-based remote monitoring system for nuclear power plant[C]//Information Science and Technology (ICIST), 2014 4th IEEE International Conference on. IEEE, 2014: 196-199.

[3] 李晓云，王亮，吴荔等．核电站远程监测系统：中国．专利号:201310754649.8[P]. 2013-12-31.

[4] 参与受海外高层次人才创新创业专项资金——孔雀计划（No.KQC201109050096A）和国家自然科学基金(Grant No.61271005)支持的《核电站环境与安全远程实时监测系统》项目研发。

[5] 参与深圳市互联网产业发展专项资金项目——《面向服务机器人应用的传感网关键技术研究》中的“智能家居应用系统研究”项目研发。

[6] 参与广东省机器人与智能信息技术创新科研团队的《服务机器人：基础理论、关键技术与系统开发》项目中的“老人助行机器人”项目研发。

[7] 参与深圳市科技计划《基于 IPv6 智能网关的中小型物联网无线传输理论和关键技术研究》。

致 谢

值此毕业论文杀青之际，回顾文章写作的这一段时光，感慨颇多。我要衷心地感谢我的导师宋高俊博士在此期间对我的点滴帮助。在我论文写作过程中遇到困难时，宋老师给了我许多耐心细致的指导。当我完成初稿自觉良好时，宋老师认真查阅我的论文并指出了许多的不足之处，同时指导我进行修改。我非常感激宋老师对我论文的严格把关，才让我顺利地完成了毕业论文。

另外，我还要感谢在中国科学院深圳先进技术研究院实习期间，导师和同事对我的指导和帮助。并依托智能仿生中心的机器人重点实验室的科研条件，帮助我完成了论文设计的相关实验测试。

此外，我要感谢我的师弟师妹以及我的同学们，在论文写作期间给了我很多鼓励和生活上的照顾。

最后，我要感谢各位老师在百忙之中审阅我的论文！由于本人水平有限，论文中肯定有许多不足之处，希望各位老师能够指出以帮助改进！

再次对以上所有老师、同学表示衷心地感谢！

附 录

实 习 证 明

兹证明 南昌航空大学 吴荔 同学（身份证号：360502198804153618）为中国科学院深圳先进技术研究院客座学生，于 集成所智能仿生中心 实习。

(注：此证明只限于购买火车票)

特此证明

导师签名：

（代章）

2015年3月18日

南昌航空大学硕士学位论文原创性声明

本人郑重声明：所呈交的硕士学位论文，是我个人在导师指导下，在南昌航空大学攻读硕士学位期间独立进行研究工作所取得的成果。尽我所知，论文中除已注明部分外不包含他人已发表或撰写过的研究成果。对本文的研究工作做出重要贡献的个人和集体，均已在文中作了明确地说明并表示了谢意。本声明的法律结果将完全由本人承担。

签名：_____日期：_____

南昌航空大学硕士学位论文使用授权书

本论文的研究成果归南昌航空大学所有，本论文的研究内容不得以其它单位的名义发表。本人完全了解南昌航空大学关于保存、使用学位论文的规定，同意学校保留并向有关部门送交论文的复印件和电子版本，允许论文被查阅和借阅。本人授权南昌航空大学，可以采用影印、缩印或其他复制手段保存论文，可以公布论文的全部或部分内容。同时授权中国科学技术信息研究所将本学位论文收录到《中国学位论文全文数据库》，并通过网络向社会公众提供信息服务。

（保密的学位论文在解密后适用本授权书）

签名：_____导师签名：_____日期：_____